Six networks on a universal neuromorphic computing substrate

Thomas Pfeil*†[1] Andreas Grübl†[1] Sebastian Jeltsch†[1] Eric Müller†[1]
Paul Müller†[1] Mihai A. Petrovici†[1] Michael Schmuker†[2,3]
Daniel Brüderle[1] Johannes Schemmel[1] Karlheinz Meier[1]

December 19, 2012

[1]Kirchhoff-Institute for Physics
Universität Heidelberg
Heidelberg, Germany

[2]Neuroinformatics & Theoretical Neuroscience
Freie Universität Berlin
Berlin, Germany

[3]Bernstein Center for Computational Neuroscience Berlin
Berlin, Germany

* Correspondence: Thomas Pfeil
Universität Heidelberg
Kirchhoff-Institute for Physics
Im Neuenheimer Feld 227
69120 Heidelberg, Germany
tel: +49-6221-549813
thomas.pfeil@kip.uni-heidelberg.de

† These authors contributed equally to this work. See acknowledgements for details.

Abstract

In this study, we present a highly configurable neuromorphic computing substrate and use it for emulating several types of neural networks. At the heart of this system lies a mixed-signal chip, with analog implementations of neurons and synapses and digital transmission of action potentials. Major advantages of this emulation device, which has been explicitly designed as a universal neural network emulator, are its inherent parallelism and high acceleration factor compared to conventional computers. Its configurability allows the realization of almost arbitrary network topologies and the use of widely varied neuronal and synaptic parameters. Fixed-pattern noise inherent to analog circuitry is reduced by calibration routines. An integrated development environment allows neuroscientists to operate the device without any prior knowledge of neuromorphic circuit design. As a showcase for the capabilities of the system, we describe the successful emulation of six different neural networks which cover a broad spectrum of both structure and functionality.

Keywords: accelerated neuromorphic hardware system, universal computing substrate, highly configurable, mixed-signal VLSI, spiking neural networks, soft winner-take-all, classifier, cortical model.

1. INTRODUCTION

By nature, computational neuroscience has a high demand for powerful and efficient devices for simulating neural network models. In contrast to conventional general-purpose machines based on a von-Neumann architecture, neuromorphic systems are, in a rather broad definition, a class of devices which implement particular features of biological neural networks in their physical circuit layout (Mead, 1989; Indiveri et al., 2009; Renaud et al., 2010). In order to discern more easily between computational substrates, the term *emulation* is generally used when referring to neural networks running on a neuromorphic back-end.

Several aspects motivate the neuromorphic approach. The arguably most characteristic feature of neuromorphic devices is inherent parallelism enabled by the fact that individual neural network components (essentially neurons and synapses) are physically implemented *in silico*. Due to this parallelism, scaling of emulated network models does not imply slowdown, as is usually the case for conventional machines. The hard upper bound in network size (given by the number of available components on the neuromorphic device) can be broken by scaling of the devices themselves, e.g., by wafer-scale integration (Schemmel et al., 2010) or massively interconnected chips (Merolla et al., 2011). Emulations can be further accelerated by scaling down time constants compared to biology, which is enabled by deep submicron technology (Schemmel et al., 2006, 2010; Brüderle et al., 2011). Unlike high-throughput computing with accelerated systems, real-time systems are often specialized for low power operation (e.g., Indiveri et al., 2006; Farquhar & Hasler, 2005).

However, in contrast to the unlimited model flexibility offered by conventional simulation, the network topology and parameter space of neuromorphic systems are often dedicated for predefined applications and therefore rather restricted (e.g., Serrano-Gotarredona et al., 2006; Merolla & Boahen, 2006; Akay, 2007; Chicca et al., 2007). Enlarging the configuration space always comes at the cost of hardware resources by occupying additional chip area. Consequently, the maximum network size is reduced, or the configurability of one aspect is decreased by increasing the configurability of another. Still, configurability costs can be counterbalanced by decreasing precision. This could concern the size of integration time steps (Imam et al., 2012a), the granularity of particular parameters (Pfeil et al., 2012) or fixed-pattern noise affecting various network components. At least the latter can be, to some extent, moderated through elaborate calibration methods (Neftci & Indiveri, 2010; Brüderle et al., 2011; Gao et al., 2012).

In this study, we present a user-friendly integrated development environment that can serve as a universal neuromorphic substrate for emulating different types of neural networks. Apart from almost arbitrary network topologies, this system provides a vast configuration space for neuron and synapse parameters (Schemmel et al., 2006; Brüderle et al., 2011). Reconfiguration is achieved on-chip and does not require additional support hardware. While some models can easily be transferred from software simulations to the neuromorphic substrate, others need modifications. These modifications take into account the limited hardware resources and compensate for fixed-pattern noise (Kaplan et al., 2009; Brüderle & Müller et al., 2009; Brüderle et al., 2010, 2011; Bill et al., 2010). In the following, we show six more networks emulated on our hardware system, each requiring its own hardware configuration in terms of network topology and neuronal as well as synaptic parameters.

2. THE NEUROMORPHIC SYSTEM

The central component of our neuromorphic hardware system is the neuromorphic microchip *Spikey*. It contains analog very-large-scale integration (VLSI) circuits modeling the electrical behavior of neurons and synapses (Figure 1). In such a *physical model*, measurable quantities in the neuromorphic circuitry have corresponding biological equivalents. For example, the membrane potential V_m of a neuron is modeled by the voltage over a capacitor C_m that, in turn, can be seen as a model of the capacitance of the cell membrane. In contrast to numerical approaches, dynamics of physical quantities like V_m evolve continuously in time. We designed our hardware systems to have time constants approximately 10^4 times faster than their biological counterparts allowing for high-throughput computing. This is achieved by reducing the size and hence the time constant of electrical components, which also allows for more neurons and synapses on a single chip. To avoid confusion between hardware and biological domains of time, voltages and currents, all parameters are specified in biological domains throughout this study.

2.1. THE NEUROMORPHIC CHIP

On *Spikey* (Figure 1), a VLSI version of the standard leaky integrate-and-fire (LIF) neuron model with conductance-based synapses is implemented (Dayan & Abbott, 2001):

$$C_\mathrm{m}\frac{\mathrm{d}V_\mathrm{m}}{\mathrm{d}t} = g_\mathrm{l}(V_\mathrm{m} - E_\mathrm{l}) + \sum_i g_i(V_\mathrm{m} - E_i) \qquad (1)$$

For its hardware implementation see Figure 1, Schemmel et al. (2006) and Indiveri et al. (2011).

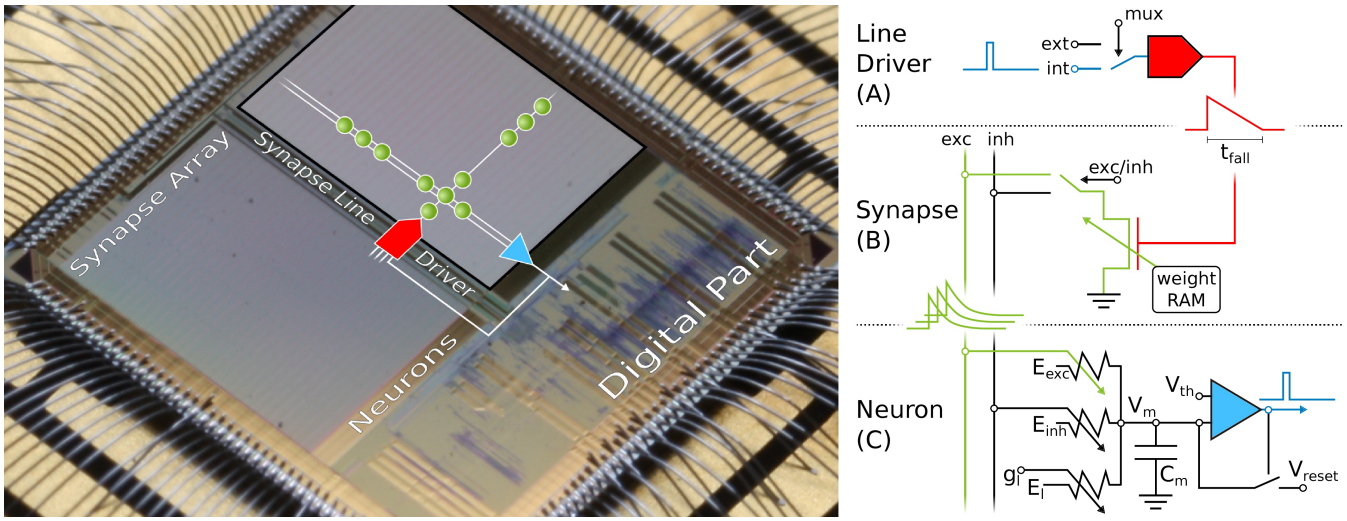

FIGURE 1: Microphotograph of the *Spikey* chip (fabricated in a $180\,\mathrm{nm}$ CMOS process with die size $5 \times 5\,\mathrm{mm}^2$). Each of its 384 neurons can be arbitrarily connected to any other neuron. In the following, we give a short overview of the technical implementation of neural networks on the *Spikey* chip. **(A)** Within the synapse array 256 synapse line drivers convert incoming digital spikes (blue) into a linear voltage ramp (red) with a falling slew rate t_{fall}. For simplicity, the slew rate of the rising edge is not illustrated here, because it is chosen very small for all emulations in this study. Each of these synapse line drivers are individually driven by either another on-chip neuron (int), e.g., the one depicted in (C), or an external spike source (ext). **(B)** Within the synapse, depending on its individually configurable weight w_i, the linear voltage ramp (red) is then translated into a current pulse (green) with exponential decay. These postsynaptic pulses are sent to the neuron via the excitatory (exc) and inhibitory (inh) input line, shared by all synapses in that array column. **(C)** Upon reaching the neuron circuit, the total current on both input lines is converted into conductances, respectively. If the membrane potential V_{m} crosses the firing threshold V_{th}, a digital pulse (blue) is generated, which can be recorded and fed back into the synapse array. After any spike, V_{m} is set to V_{reset} for a refractory time period of τ_{refrac}. Neuron and synapse line driver parameters can be configured as summarized in Table 1.

Synaptic conductances g_i (with the index i running over all synapses) drive the membrane potential V_{m} towards the reversal potential E_i, with $E_i \in \{E_{\mathrm{exc}}, E_{\mathrm{inh}}\}$. The time course of the synaptic activation is modeled by

$$g_i(t) = p_i(t) \cdot w_i \cdot g_i^{\max} \tag{2}$$

where $g_i^{\max}$ are the maximum conductances and w_i the weights for each synapse, respectively. The time course $p_i(t)$ of synaptic conductances is a linear transformation of the current pulses shown in Figure 1 (green), and hence an exponentially decaying function of time. The generation of conductances at the neuron side is described in detail by Indiveri et al. (2011), postsynaptic potentials are measured by Schemmel et al. (2007).

The implementation of spike-timing dependent plasticity (STDP; Bi & Poo, 1998; Song et al., 2000) modulating w_i over time is described in Schemmel et al. (2006) and Pfeil et al. (2012). Correlation measurement between pre- and post-synaptic action potentials is carried out in each synapse, and the 4-bit weight is updated by an on-chip controller located in the digital part of the *Spikey* chip. However, STDP will not be further discussed in this study.

Short-term plasticity (STP) modulates $g_i^{\max}$ (Schemmel et al., 2007) similar to the model by Tsodyks & Markram (1997) and Markram et al. (1998). On hardware, STP can be configured individually for each synapse line driver that corresponds to an axonal connection in biological terms. It can either be facilitating or depressing.

The propagation of spikes within the *Spikey* chip is illustrated in Figure 1 and described in detail by Schemmel et al. (2006). *Spikes* enter the chip as time-stamped events using standard digital signaling techniques that facilitate long-range communication, e.g., to the host computer or other chips. Such digital packets are processed in discrete time in the digital part of the chip, where they are transformed into digital *pulses* entering the synapse line driver (blue in Figure 1A). These pulses propagate in continuous time between on-chip neurons, and are optionally transformed back into digital spike packets for off-chip communication.

2.2. SYSTEM ENVIRONMENT

The *Spikey* chip is mounted on a network module schematized in Figure 2 and described in Fieres et al. (2004). Digital spike and configuration data is transferred

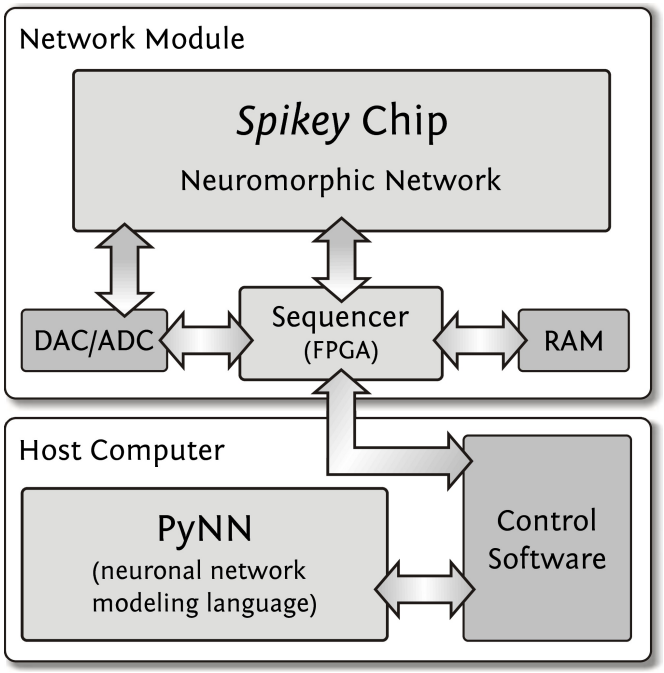

FIGURE 2: Integrated development environment: User access to the *Spikey* chip is provided using the `PyNN` neural network modeling language. The control software controls and interacts with the network module which is operating the *Spikey* chip. The RAM size (512 MB) limits the total number of spikes for stimulus and spike recordings to approx. $2 \cdot 10^8$ spikes. The required data for a full configuration of the *Spikey* chip has a size of approx. 100 kB.

via direct connections between a field-programmable gate array (FPGA) and the *Spikey* chip. Onboard digital-to-analog converter (DAC) and analog-to-digital converter (ADC) components supply external parameter voltages to the *Spikey* chip and digitize selected voltages generated by the chip for calibration purposes. Furthermore, up to eight selected membrane voltages can be recorded in parallel by an oscilloscope. Because communication between a host computer and the FPGA has a limited bandwidth that does not satisfy real-time operation requirements of the *Spikey* chip, experiment execution is controlled by the FPGA while operating the *Spikey* chip in continuous time. To this end, all experiment data is stored in the local random access memory (RAM) of the network module. Once the experiment data is transferred to the local RAM, emulations run with an acceleration factor of 10^4 compared to biological real-time. This acceleration factor applies to all emulations shown in this study, independent of the size of networks.

Execution of an experiment is split up into three steps (Figure 2). First, the *control software* within the memory of the host computer generates configuration data (Table 1, e.g., synaptic weights, network connectivity, etc.), as well as input stimuli to the network. All data is stored as a sequence of commands and is transferred to the memory on the network module. In the second step, a playback sequencer in the FPGA logic interprets this data and sends it to the *Spikey* chip, as well as triggers the emulation. Data produced by the chip, e.g., neuronal activity in terms of spike times, is recorded in parallel. In the third and final step, this recorded data stored in the memory on the network module are retrieved and transmitted to the host computer, where they are processed by the control software.

Having a control software that abstracts hardware greatly simplifies modeling on the neuromorphic hardware system. However, modelers are already struggling with multiple incompatible interfaces to software simulators. That is why our neuromorphic hardware system supports `PyNN`, a widely used application programming interface (API) that strives for a coherent user interface, allowing portability of neural network models between different software simulation frameworks (e.g., `NEST` or `NEURON`) and hardware systems (e.g., the *Spikey* system). For details see Gewaltig & Diesmann (2007); Eppler et al. (2009) for `NEST`, Carnevale & Hines (2006); Hines et al. (2009) for `NEURON`, Brüderle et al. (2011); Brüderle & Müller et al. (2009) for the *Spikey* chip, and Davison et al. (2009, 2010) for `PyNN`, respectively.

2.3. CONFIGURABILITY

In order to facilitate the emulation of network models inspired by biological neural structures, it is essential to support the implementation of different (cortical) neuron types. From a mathematical perspective, this can be achieved by varying the appropriate parameters of the implemented neuron model (Equation 1).

To this end, the *Spikey* chip provides 2969 different analog parameters (Table 1) stored on current memory cells that are continuously refreshed from a digital on-chip memory. Most of these cells deliver individual parameters for each neuron (or synapse line driver), e.g., leakage conductances g_l. Due to the size of the current-voltage conversion circuitry it was not possible to provide individual voltage parameters, such as, e.g., E_l, E_exc and E_inh, for each neuron. As a consequence, groups of 96 neurons share most of these voltage parameters. Parameters that can not be controlled individually are delivered by global current memory cells.

In addition to the possibility of controlling analog parameters, the *Spikey* chip also offers an almost arbitrary configurability of the network topology. As illustrated in Figure 1, the fully configurable *synapse array* allows connections from synapse line drivers (located alongside the array) to arbitrary neurons (located below the array)

Scope	Name	Type	Description
Neuron circuits (A)	n/a	$\mathsf{i_n}$	Two digital configuration bits activating the neuron and readout of its membrane voltage
	g_l	$\mathsf{i_n}$	Bias current for neuron leakage circuit
	τ_refrac	$\mathsf{i_n}$	Bias current controlling neuron refractory time
	E_l	$\mathsf{s_n}$	Leakage reversal potential
	E_inh	$\mathsf{s_n}$	Inhibitory reversal potential
	E_exc	$\mathsf{s_n}$	Excitatory reversal potential
	V_th	$\mathsf{s_n}$	Firing threshold voltage
	V_reset	$\mathsf{s_n}$	Reset potential
Synapse line drivers (B)	n/a	$\mathsf{i_l}$	Two digital configuration bits selecting input of line driver
	n/a	$\mathsf{i_l}$	Two digital configuration bits setting line excitatory or inhibitory
	$t_\mathrm{rise}, t_\mathrm{fall}$	$\mathsf{i_l}$	Two bias currents for rising and falling slew rate of presynaptic voltage ramp
	g_i^max	$\mathsf{i_l}$	Bias current controlling maximum voltage of presynaptic voltage ramp
Synapses (B)	w	$\mathsf{i_s}$	4-bit weight of each individual synapse
STP related (C)	n/a	$\mathsf{i_l}$	Two digital configuration bits selecting short-term depression or facilitation
	U_SE	$\mathsf{i_l}$	Two digital configuration bits tuning synaptic efficacy for STP
	n/a	$\mathsf{s_l}$	Bias voltage controlling spike driver pulse length
	τ_rec, τ_facil	$\mathsf{s_l}$	Voltage controlling STP time constant
	I	$\mathsf{s_l}$	Short-term facilitation reference voltage
	R	$\mathsf{s_l}$	Short-term capacitor high potential
STDP related (D)	n/a	$\mathsf{i_l}$	Bias current controlling delay for presynaptic correlation pulse (for calibration purposes)
	$A_{+/-}$	$\mathsf{s_l}$	Two voltages dimensioning charge accumulation per (anti-)causal correlation measurement
	n/a	$\mathsf{s_l}$	Two threshold voltages for detection of relevant (anti-)causal correlation
	τ_STDP	g	Voltage controlling STDP time constants

TABLE 1: List of analog current and voltage parameters as well as digital configuration bits. Each with corresponding model parameter names, excluding technical parameters that are only relevant for correctly biasing analog support circuitry or controlling digital chip functionality. Electronic parameters that have no direct translation to model parameters are denoted *n/a*. The membrane capacitance is fixed and identical for all neuron circuits ($C_\mathrm{m} = 0.2\,\mathrm{nF}$ in biological value domain). Parameter types: (i) controllable for each corresponding circuit: 192 for neuron circuits (denoted with subscript n), 256 for synapse line drivers (denoted with subscript l), 49152 for synapses (denoted with subscript s), (s) two values, shared for all even/odd neuron circuits or synapse line drivers, respectively, (g) global, one value for all corresponding circuits on the chip. All numbers refer to circuits associated to one synapse array and are doubled for the whole chip. For technical reasons, the current revision of the chip only allows usage of one synapse array of the chip. Therefore, all experiments presented in this paper are limited to a maximum of 192 neurons. For parameters denoted by (A) see Equation 1 and Schemmel et al. (2006), for (B) see Figure 1, Equation 2 and Dayan & Abbott (2001), for (C) see Schemmel et al. (2007) and for (D) see Schemmel et al. (2006) and Pfeil et al. (2012).

via synapses whose weights can be set individually with a 4-bit resolution. This limits the maximum fan-in to 256 synapses per neuron, which can be composed of up to 192 synapses from on-chip neurons, and up to 256 synapses from external spike sources. Because the total number of neurons exceeds the number of inputs per neuron, an all-to-all connectivity is not possible. For all networks presented in this study, connectivity is sparser than is available on the chip, which supports the chosen trade-off between inputs per neuron and total neuron count.

2.4. CALIBRATION

Device mismatch that arises from hardware production variability causes fixed-pattern noise, which causes parameters to vary from neuron to neuron as well as from synapse to synapse. Electronic noise (including thermal noise) also affects dynamic variables, as, e.g., the membrane potential V_m. Consequently, experiments will exhibit some amount of both neuron-to-neuron and trial-to-trial variability given the same input stimulus. It is, however, important to note that these types of variations are not unlike the neuron diversity and response stochasticity found in biology (Gupta et al., 2000; Maass et al., 2002; Marder & Goaillard, 2006; Rolls & Deco, 2010).

To facilitate modeling and provide repeatability of experiments on arbitrary *Spikey* chips, it is essential to minimize these effects by calibration routines. Many calibrations have directly corresponding biological model parameters, e.g., membrane time constants (described in the following), firing thresholds, synaptic efficacies or PSP

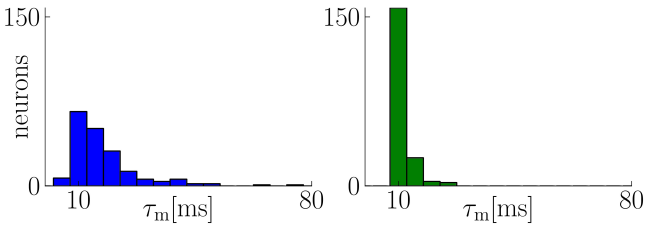

FIGURE 3: Before calibration (left), the distribution of τ_m values has a median of $\widetilde{\tau_m} = 15.1\,\text{ms}$ with 20th and 80th percentiles of $\tau_m^{20} = 10.3\,\text{ms}$ and $\tau_m^{80} = 22.1\,\text{ms}$, respectively. After calibration (right), the distribution median lies closer to the target value and narrows significantly: $\widetilde{\tau_m} = 11.2\,\text{ms}$ with $\tau_m^{20} = 10.6\,\text{ms}$ and $\tau_m^{80} = 12.0\,\text{ms}$. Two neurons were discarded, because the automated calibration algorithm did not converge.

shapes. Others have no equivalents, like compensations for shared parameters or workarounds of defects (e.g., Kaplan et al., 2009; Bill et al., 2010; Pfeil et al., 2012). In general, calibration results are used to improve the mapping between biological input parameters and the corresponding target hardware voltages and currents, as well as to determine the dynamic range of all model parameters (e.g., Brüderle & Müller et al., 2009).

While the calibration of most parameters is rather technical, but straightforward (e.g., all neuron voltage parameters), some require more elaborate techniques. These include the calibration of τ_m, STP as well as synapse line drivers, as we describe later for individual network models. The membrane time constant $\tau_m = C_m/g_l$ differs from neuron to neuron mostly due to variations in the leakage conductance g_l. However, g_l is independently adjustable for every neuron. Because this conductance is not directly measurable, an indirect calibration method is employed. To this end, the threshold potential is set below the resting potential. Following each spike, the membrane potential is clamped to V_{reset} for an absolute refractory time τ_{refrac}, after which it evolves exponentially towards the resting potential E_l until the threshold voltage triggers a spike and the next cycle begins. If the threshold voltage is set to $V_{th} = E_l - 1/e \cdot (E_l - V_{reset})$, the spike frequency equals $1/(\tau_m + \tau_{refrac})$, thereby allowing an indirect measurement and calibration of E_l and therefore τ_m. For a given τ_m and $\tau_{refrac} = const$, V_{th} can be calculated. An iterative method is applied to find the best-matching V_{th}, because the exact hardware values for E_l, V_{reset} and V_{th} are only known after the measurement. The effect of calibration on a typical chip can best be exemplified for a typical target value of $\tau_m = 10\,\text{ms}$. Figure 3 depicts the distribution of τ_m of a typical chip before and after calibration.

The STP hardware parameters have no direct translation to model equivalents. In fact, the implemented transconductance amplifier tends to easily saturate within the available hardware parameter ranges. These non-linear saturation effects can be hard to handle in an automated fashion on an individual circuit basis. Consequently, the translation of these parameters is based on STP courses averaged over several circuits.

3. HARDWARE EMULATION OF NEURAL NETWORKS

In the following, we present six neural network models that have been emulated on the *Spikey* chip. Most of the emulation results are compared to those obtained by software simulations in order to verify the network functionality and performance. For all these simulations the tool `NEST` (Gewaltig & Diesmann, 2007) or `NEURON` (Carnevale & Hines, 2006) is used.

3.1. SYNFIRE CHAIN WITH FEEDFORWARD INHIBITION

Architectures with a feedforward connectivity have been employed extensively as computational components and as models for the study of neuronal dynamics. Synfire chains are feedforward networks consisting of several neuron groups where each neuron in a group projects to neurons in the succeeding group.

They have been originally proposed to account for the presence of behaviorally-related, highly precise firing patterns (Baker et al., 2001; Prut et al., 1998). Further properties of such structures have been studied extensively, including activity transport (Aertsen et al., 1996; Diesmann et al., 1999; Litvak et al., 2003), external control of information flow (Kremkow et al., 2010), computational capabilities (Abeles et al., 2004; Vogels & Abbott, 2005; Schrader et al., 2010), complex dynamic behavior (Yazdanbakhsh et al., 2002) and their embedding into surrounding networks (Aviel et al., 2003; Tetzlaff et al., 2005; Schrader et al., 2008). Kremkow et al. (2010) have shown that feedforward inhibition can increase the selectivity to the initial stimulus and that the local delay of inhibition can modify this selectivity.

3.1.1. Network Topology

The presented network model is an adaptation of the *feedforward network* described in Kremkow et al. (2010).

The network consists of several neuron groups, each comprising $n_{RS} = 100$ excitatory regular spiking (RS) and $n_{FS} = 25$ inhibitory fast spiking (FS) cells. All neurons are modeled as LIF neurons with exponentially decaying synaptic conductance courses. According to Kremkow et al. (2010) all neurons have identical parameters.

As shown in Figure 4A, RS neurons project to both RS and FS populations in the subsequent group while the FS population projects to the RS population in its local

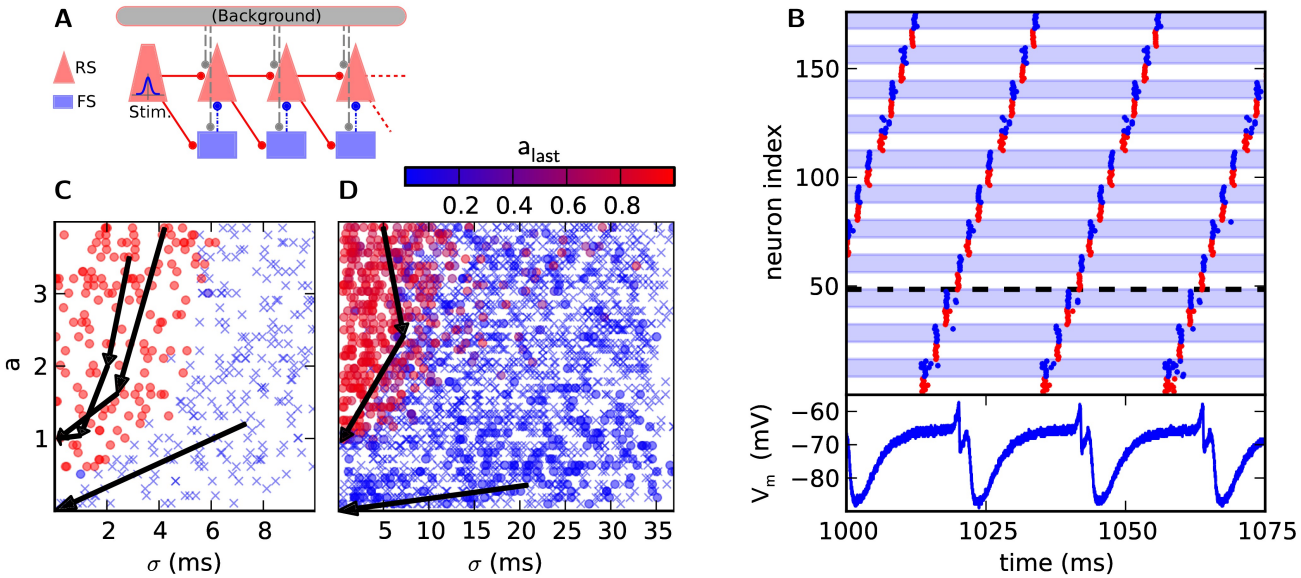

FIGURE 4: (A) Synfire chain with feedforward inhibition. The background is only utilized in the original model, where it is implemented as random Gaussian current. For the presented hardware implementation it has been omitted due to network size constraints. As compensation for missing background stimuli, the resting potential was increased to ensure a comparable excitability of the neurons. **(B)** Hardware emulation. Top: Raster plot of pulse packet propagation $1000\,\mathrm{ms}$ after initial stimulus. Spikes from RS groups are shown in red and spikes from FS groups are denoted by blue color and background. Bottom: Membrane potential of the first neuron in the fourth RS group, which is denoted by a dashed horizontal line. The cycle duration is approximately $20\,\mathrm{ms}$. **(C)** State space generated with software simulations of the original model. The position of each marker indicates the (σ, a) parameters of the stimulus while the color encodes the activity in the RS population of the third synfire group. Lack of activity is indicated with a cross. The evolution of the pulse packet parameters is shown for three selected cases by a series of arrows. Activity either stably propagates with fixed point $(\sigma, a) = (0.1\,\mathrm{ms}, 1)$ or extinguishes with fixed point $(\sigma, a) = (0\,\mathrm{ms}, 0)$. **(D)** Same as (C), but emulated on the FACETS chip-based system. The activity in the last group is located either near $(\sigma, a) = (0\,\mathrm{ms}, 0)$ or $(0.3\,\mathrm{ms}, 1)$. The difference to software simulations is explained in Section 3.1.2.

group. Each neuron receives a fixed number of randomly chosen inputs from each presynaptic population. The first group is stimulated by a population of n_{RS} external spike sources with identical connection probabilities as used for RS groups within the chain.

Two different criteria are employed to assess the functionality of the emulated synfire chain. The first, straightforward benchmark is the stability of signal propagation. An initial synchronous stimulus is expected to cause a stable propagation of activity, with each neuron in an RS population spiking exactly once. Deviations from the original network parameters can cause the activity to grow rapidly, i.e., each population emits more spikes than its predecessor, or stall pulse propagation.

The second, broader characterization follows Kremkow et al. (2010), who has analyzed the response of the network to various stimuli. The stimulus is parametrized by the variables a and σ. For each neuron in the stimulus population a spike times are generated by sampling them from a Gaussian distribution with common mean and standard deviation. σ is defined as the standard deviation of the spike times of all source neurons. Spiking activity that is evoked in the subsequent RS populations is characterized analogously by measuring a and σ.

Figure 4C shows the result of a software simulation of the original network. The filter properties of the network are reflected by a separatrix dividing the state space shown in Figure 4C and D into two areas, each with a different fixed point. First, the basin of attraction (dominated by red circles in Figure 4C) from which stable propagation can be evoked and second, the remaining region (dominated by crosses in Figure 4C) where any initial activity becomes extinguished. This separatrix determines which types of initial input lead to a stable signal propagation.

3.1.2. *Hardware Emulation*

The original network model could not be mapped directly to the *Spikey* chip because it requires 125 neurons per group, while the chip accommodates 192 neuron circuits. Further constraints were caused by the fixed synaptic delays, which are determined by the speed of signal

propagation on the chip. The magnitude of the delay is approximately 1 ms in biological time.

By simple modifications of the network, we were able to qualitatively reproduce both benchmarks defined in Section 3.1.1. Two different network configurations were used, each adjusted to the requirements of one benchmark. In the following, we describe these differences, as well as the results for each benchmark.

To demonstrate a stable propagation of pulses, a large number of consecutive group activations was needed. The chain was configured as a loop by connecting the last group to the first, allowing the observation of more pulse packet propagations than there are groups in the network.

The time between two passes of the pulse packet at the same synfire group needs to be maximized to allow the neurons to recover (see voltage trace in Figure 4B). This is accomplished by increasing the group count and consequently reducing the group size. As too small populations cause an unreliable signal propagation, which is mainly caused by inhomogeneities in the neuron behavior, $n_{\mathrm{RS}} = n_{\mathrm{FS}} = 8$ was chosen as a satisfactory trade-off between propagation stability and group size. Likewise, the proportion of FS neurons in a group was increased to maintain a reliable inhibition. To further improve propagation properties, the membrane time constant was lowered for all neurons by raising g_{l} to its maximum value. The strength of inhibition was increased by setting the inhibitory synaptic weight to its maximum value and lowering the inhibitory reversal potential to its minimum value. Finally, the synaptic weights $RS_{\mathrm{i}} \to RS_{\mathrm{i}+1}$ and $RS_{\mathrm{i}} \to FS_{\mathrm{i}+1}$ were adjusted. With these improvements we could observe persisting synfire propagation on the oscilloscope 2 h wall-clock time after stimulation. This corresponds to more than 2 years in biological real-time.

The second network demonstrates the filtering properties of a hardware-emulated synfire chain with feedforward inhibition. This use case required larger synfire groups than in the first case as otherwise, the total excitatory conductance caused by a pulse packet with large σ was usually not smooth enough due to the low number of spikes. Thus, three groups were placed on a single chip with $n_{\mathrm{RS}} = 45$ and $n_{\mathrm{FS}} = 18$. The resulting evolution of pulse packets is shown in Figure 4D. After passing three groups, most runs resulted in either very low activity in the last group or were located near the point $(0.3\,\mathrm{ms}, 1)$, as illustrated in Figure 4D.

Emulations on hardware differ from software simulations in two important points: First, the separation in the parameter space of the initial stimulus is not as sharply bounded, which is demonstrated by the fact that occasionally, significant activity in the last group can be evoked by stimuli with large σ and large a, as seen in Figure 4D. This is a combined effect due to the reduced population sizes and the fixed pattern noise in the neuronal and synaptic circuits. Second, a stimulus with a small a can evoke weak activity in the last group, which is attributed to a differing balance between excitation and inhibition. In hardware, a weak stimulus causes both, the RS and FS populations to response weakly which leads to a weak inhibition of the RS population, allowing the pulse to reach the last synfire group. Hence, the pulse fades slowly instead of being extinguished completely. In the original model, the FS population is more responsive and prevents the propagation more efficiently.

Nevertheless, the filtering properties of the network are apparent. The quality of the filter could be improved by employing the original group size, which would require using a large-scale neuromorphic device (see, e.g., Schemmel et al., 2010).

Our hardware implementation of the synfire chain model demonstrates the possibility to run extremely long lasting experiments due to the high acceleration factor of the hardware system. Because the synfire chain model itself does not require sustained external stimulus, it could be employed as an autonomous source of periodic input to other experiments.

3.2. BALANCED RANDOM NETWORK

Brunel (2000) reports *balanced random networks* (BRNs) exhibiting, among others, asynchronous irregular network states with stationary global activity.

3.2.1. Network Topology

BRNs consist of an inhibitory and excitatory population of neurons, both receiving feedforward connections from two populations of Poisson processes mimicking background activity. Both neuron populations are recurrently connected including connections within a population. All connections are realized with random and sparse connections of probability p. In this study, synaptic weights for inhibitory connections are chosen four times larger than those for excitatory ones. In contrast to the original implementation using 12500 neurons, we scaled this network by a factor of 100 while preserving its firing behavior.

If single cells fire irregularly, the *coefficient of variation*

$$C_{\mathrm{V}} = \frac{\sigma_{\mathrm{T}}}{\overline{T}} \tag{3}$$

of interspike intervals has values close to or higher than one (Dayan & Abbott, 2001). $\overline{T}$ and σ_{T} are the mean and standard deviation of these intervals. Synchrony between two cells can be measured by calculating the *correlation coefficient*

$$C_{\mathrm{C}} = \frac{\mathrm{cov}(n_1, n_2)}{\sqrt{\mathrm{var}(n_1)\mathrm{var}(n_2)}} \tag{4}$$

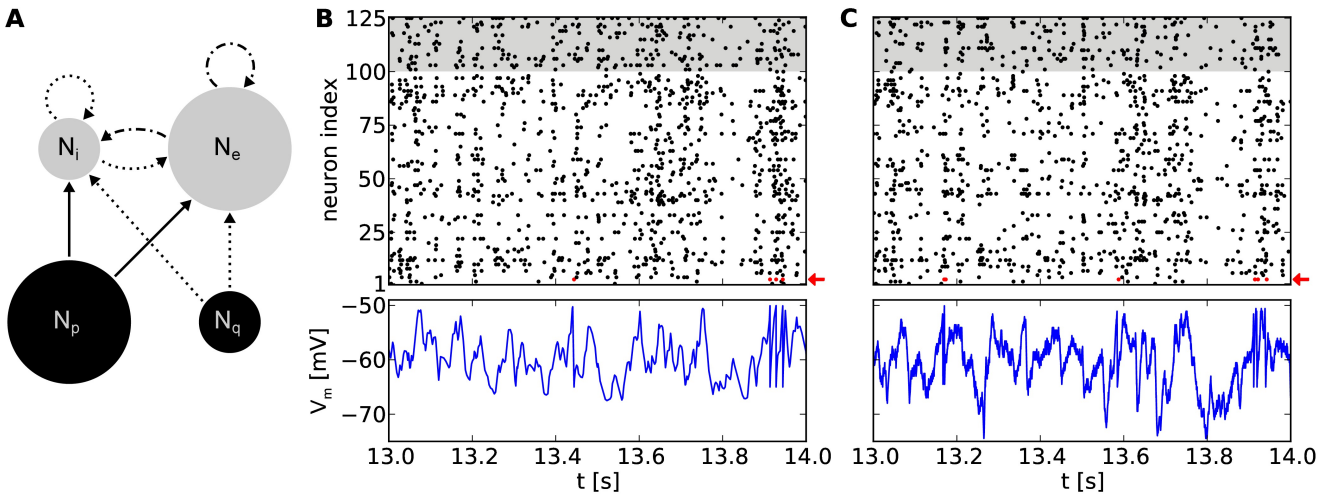

FIGURE 5: Network topology of a balanced random network. **(A)** Populations consisting of $N_\mathrm{e} = 100$ excitatory and $N_\mathrm{i} = 25$ inhibitory neurons (gray circles), respectively, are stimulated by populations of Poisson sources (black circles). We use $N_\mathrm{p} = 100$ independent sources for excitation and $N_\mathrm{q} = 25$ for inhibition. Arrows denote projections between these populations with connection probabilities $p = 0.1$, with solid lines for excitatory and dotted lines for inhibitory connections. Dot and dash lines are indicating excitatory projections with short-term depression. **(B)** Top: Raster plot of a software simulation. Populations of excitatory and inhibitory neurons are depicted with white and gray background, respectively. Note that for clarity only the time interval [13 s..14 s] of a 20 s emulation is shown. For the full 20 s emulation, we have measured $C_\mathrm{V} = 0.96 \pm 0.09$ (mean over all neurons) and $C_\mathrm{C} = 0.010 \pm 0.017$ (mean over 1000 random chosen pairs of neurons), respectively. Bottom: Recorded membrane potential of an arbitrary excitatory neuron (neuron index 3, highlighted with a red arrow in the above raster plot). **(C)** Same network topology and stimulus as in (B), but emulated on the *Spikey* chip, resulting in $C_\mathrm{V} = 1.02 \pm 0.16$ and $C_\mathrm{C} = 0.014 \pm 0.019$. Note that the membrane recordings are calibrated such that the threshold and reset potential match those of the software counterpart.

of their spike trains n_1 and n_2, respectively (Perkel et al., 1967). The variance (var) and covariance (cov) are calculated by using time bins with 2 ms duration (Kumar et al., 2008).

Brüderle et al. (2010) have shown another approach to investigate networks inspired by Brunel (2000). Their focus have been the effects of network parameters and STP on the firing rate of the network. In our study, we show that such BRNs can show an asynchronous irregular network state, when emulated on hardware.

3.2.2. *Hardware Emulation*

In addition to standard calibration routines (Section 2.4), we have calibrated the chip explicitly for the BRN shown in Figure 5A. In the first of two steps, excitatory and inhibitory synapse line drivers were calibrated sequentially towards equal strength, respectively, but with inhibition four times stronger than excitation. To this end, all available neurons received spiking activity from a single synapse line driver, thereby averaging out neuron-to-neuron variations. The shape of synaptic conductances (specifically t_fall and g_i^max) were adjusted to obtain a target mean firing rate of 10 Hz over all neurons. Similarly, each driver was calibrated for its inhibitory operation mode. All neurons were strongly stimulated by an additional driver with its excitatory mode already calibrated, and again the shape of conductances, this time for inhibition, was adjusted to obtain the target rate.

Untouched by this prior calibration towards a target mean rate, neuron excitability still varied between neurons and was calibrated consecutively for each neuron in a second calibration step. For this, all neurons of the BRN were used to stimulate a single neuron with a total firing rate that was uniformly distributed among all inputs and equal to the estimated firing rate of the final network implementation. Subsequently, all afferent synaptic weights to this neuron were scaled in order to adapt its firing rate to the target rate.

To avoid a self-reinforcement of network activity observed in emulations on the hardware, efferent connections of the excitatory neuron population were modeled as short-term depressing. Nevertheless, such BRNs still show an asynchronous irregular network state (Figure 5B).

Figure 5C show recordings of a BRN emulation on a calibrated chip with neurons firing irregularly and asynchronously. Note that $C_\mathrm{V} \geq 1$ does not necessarily guarantee an exponential interspike interval distribution and

even less Poisson firing. However, neurons within the BRN clearly exhibit irregular firing (compare raster plots of Figure 5B and C).

A simulation of the same network topology and stimulus using software tools produced similar results. Synaptic weights were not known for the hardware emulation, but defined by the target firing rates using the above calibration. A translation to biological parameters is possible, but would have required further measurements and was not of further interest in this context. Instead, for software simulations, the synaptic weight for excitatory connections were chosen to fit the mean firing rate of the hardware emulation (approx. 9 Hz). Then, the weight of inhibitory connections were chosen to preserve the ratio between inhibitory and excitatory weights.

Membrane dynamics of single neurons within the network are comparable between hardware emulations and software simulations (Figure 5B and C). Evidently, spike times differ between the two approaches due to various hardware noise sources (Section 2.4). However, in "large" populations of neurons ($Ne + Ni = 125$ neurons), we observe that these phenomena have qualitatively no effect on firing statistics, which are comparable to software simulations (compare raster plots of Figure 5B and C). The ability to reproduce these statistics is highly relevant in the context of cortical models which rely on asynchronous irregular firing activity for information processing (e.g., van Vreeswijk & Sompolinsky, 1996).

3.3. SOFT WINNER-TAKE-ALL NETWORK

Soft winner-take-all (sWTA) computation is often viewed as an underlying principle in models of cortical processing (Grossberg, 1973; Maass, 2000; Itti & Koch, 2001; Douglas & Martin, 2004; Oster et al., 2009; Lundqvist et al., 2010). The sWTA architecture has many practical applications, for example contrast enhancement, or making a decision which of two concurrent inputs is larger. Many neuromorphic systems explicitly implement sWTA architectures (Lazzaro et al., 1988; Chicca et al., 2007; Neftci & Indiveri, 2010).

3.3.1. Network Topology

We implemented an sWTA network that is composed of a ring-shaped layer of recurrently connected excitatory and a common pool of inhibitory neurons (Figure 6A), following the implementation by Neftci et al. (2011). 50 excitatory neurons project to the common inhibitory pool of 16 neurons and receive recurrent feedback from there. In addition, excitatory neurons have recurrent excitatory connections to their neighbors on the ring. The strength of these decays with increasing distance on the ring, following a Gaussian profile with a standard deviation of $\sigma_{\text{rec}} = 5$ neurons. External stimulation is also received through a Gaussian profile, with the mean μ_{ext} expressing the neuron index that receives input with maximum synaptic strength. Synaptic input weights to neighbors of that neuron decay according to the standard deviation of $\sigma_{\text{ext}} = 3$ neurons. We clipped the input weights to zero beyond $\sigma_{\text{ext}} \cdot 3$. Each neuron located within the latter Gaussian profile receives stimulation from five independent Poisson spike sources each firing at rate r. Depending on the contrast between the input firing rates r_1 and r_2 of two stimuli applied to opposing sides of the ring, one side of the ring "wins" by firing with a higher rate and thereby suppressing the other.

3.3.2. Hardware Emulation

We assessed the efficiency of the sWTA circuit by measuring the reduction in firing rate exerted in neurons when the opposite side of the ring is stimulated. We stimulated one side of the ring (μ_{ext} = neuron index 13) with a constant firing rate of $r_1 = 50$ Hz. The opposite of the ring (μ_{ext} = neuron index 38) was stimulated with varying firing rate r_2 between zero and 100 Hz. In case of hardware emulations, each stimulus was distributed and hence averaged over multiple line drivers in order to equalize stimulation strength among neurons. For both back-ends, inhibitory weights were chosen four times stronger than excitatory ones (using the synapse line driver calibration of Section 3.2).

We measured the average firing rate r_{tot} of all neurons in each half of the ring, averaging over 5 runs with 2 s duration and different random number seeds. The firing rate of the reference side decreased when the firing rate of stimulation to the opposite side was increased, both in software simulation and on the hardware (Figure 6B and C). In both cases, the average firing rates crossed at $r_2 = 50$ Hz, corresponding to the spike rate delivered to the reference side. The ratio between firing rates r_{tot} (ratio between gray and black trace) is smaller for hardware emulations compared to software simulations, but still sufficient to produce robust sWTA functionality. Note that the observed firing rates are higher on the hardware than in the software simulation. This difference is due to the fact that we had to increase synaptic weights on the hardware in order to operate it at its working point in terms of network activity. The higher firing rates significantly improved the reliability of the network performance.

Figure 6D and E depict activity profiles of the neuron layers illustrating recurrent excitation. In both cases, the rate profile is sharpened compared to the comparably broad input profile. The hardware neurons exhibited a broader and also slightly asymmetric excitation profile compared to the software simulation. The asymmetry is likely due to inhomogeneity of excitability due to device

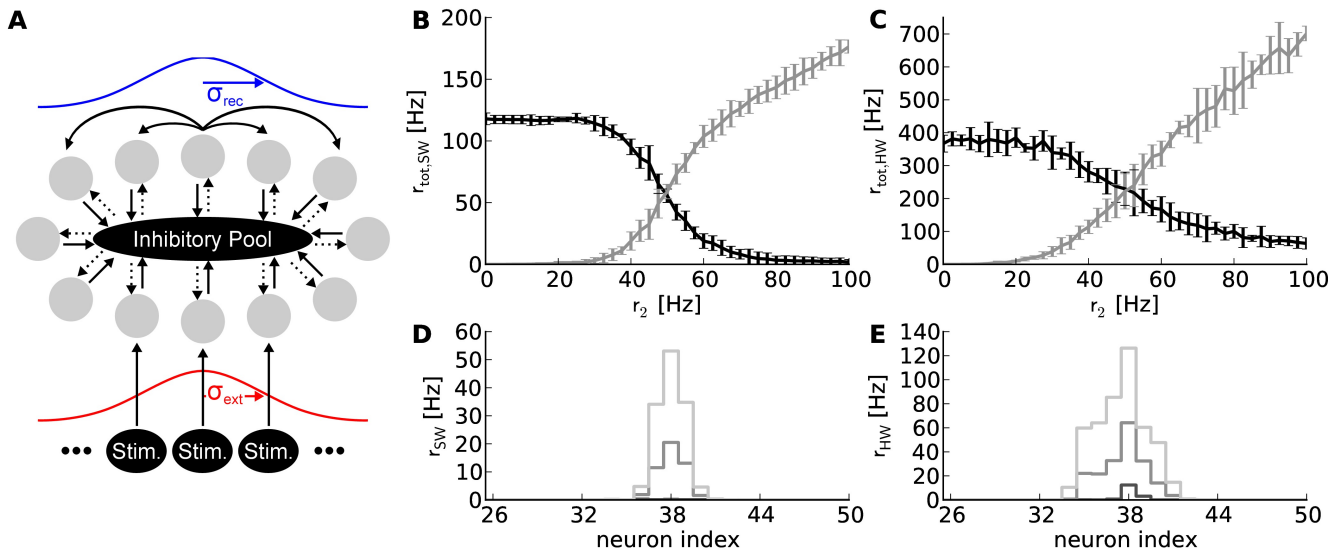

FIGURE 6: (A) Topology of a soft winner-take-all network. Gray circles: excitatory neurons. Solid arrows: excitatory, dotted arrows: inhibitory connections. Blue curve: strength profile of recurrent connections between excitatory neurons. Red curve: strength profile for external stimulations. For details see text. **(B)** Results of software simulation (SW). Black curve: Average firing rate of the reference half where constant external stimulation ($r_1 = 50\,\mathrm{Hz}$) is received. Gray curve: Average firing rate of the neurons in the half of the ring where varying external stimulation with rate r_2 is received. **(C)** Same network topology and stimulus as (B), but emulated on *Spikey* (HW). **(D)** Firing rate distribution over neuron indices for $r_2 = 25\,\mathrm{Hz}$ (black), $50\,\mathrm{Hz}$ (dark gray) and $75\,\mathrm{Hz}$ (light gray). **(E)** Same as (D), but emulated on *Spikey*.

mismatch (Section 2). The broader excitation profile indicates that inhibitition is less efficient on the hardware than in the software simulation (a trend that can also be observed in the firing rates in Figure 6B and C). Counteracting this loss of inhibition may be possible through additional calibration, if the sharpness of the excitation profile is critical for the task in which such an sWTA circuit is to be employed.

Taken together, sWTA functionality is well reproduced on the *Spikey* chip, qualifying our hardware system for applications relying on similar sWTA network topologies.

3.4. CORTICAL LAYER 2/3 ATTRACTOR MODEL

Throughout the past decades, attractor networks that model working memory in the cerebral cortex have gained increasing support from both experimental data and computer simulations. The *cortical layer 2/3 attractor memory model* described in Lundqvist et al. (2006, 2010) has been remarkably successful at reproducing both low-level (firing patterns, membrane potential dynamics) and high level (pattern completion, attentional blink) features of cortical information processing. One particularly valuable aspect is the very low amount of fine-tuning this model requires in order to reproduce the rich set of desired internal dynamics. It has also been shown in Brüderle et al. (2011) that there are multiple ways of scaling this model down in size without affecting its main functionality features. These aspects make it an ideal candidate for implementation on our analog neuromorphic device. In this context, it becomes particularly interesting to analyze how the strong feedback loops which predominantly determine the characteristic network activity are affected by the imposed limitations of the neuromorphic substrate and fixed-pattern noise. Here, we extend the work done in Brüderle et al. (2011) by investigating specific attractor properties such as firing rates, voltage UP-states and the pattern completion capability of the network.

3.4.1. *Network Topology*

From a structural perspective, the most prominent feature of the Layer 2/3 Attractor Memory Network is its modularity. Faithful to its biological archetype, it implements a set of cortical hypercolumns, which are in turn subdivided into multiple minicolumns (Figure 7A). Each minicolumn consists of three cell populations: excitatory pyramidal cells, inhibitory basket cells and inhibitory RSNP (regular spiking non-pyramidal) cells.

Attractor dynamics arise from the synaptic connectivity on two levels. Within a hypercolumn, the basket cell population enables a soft-WTA-like competition among the pyramidal populations within the minicolumns. On a global scale, the long-range inhibition mediated by the

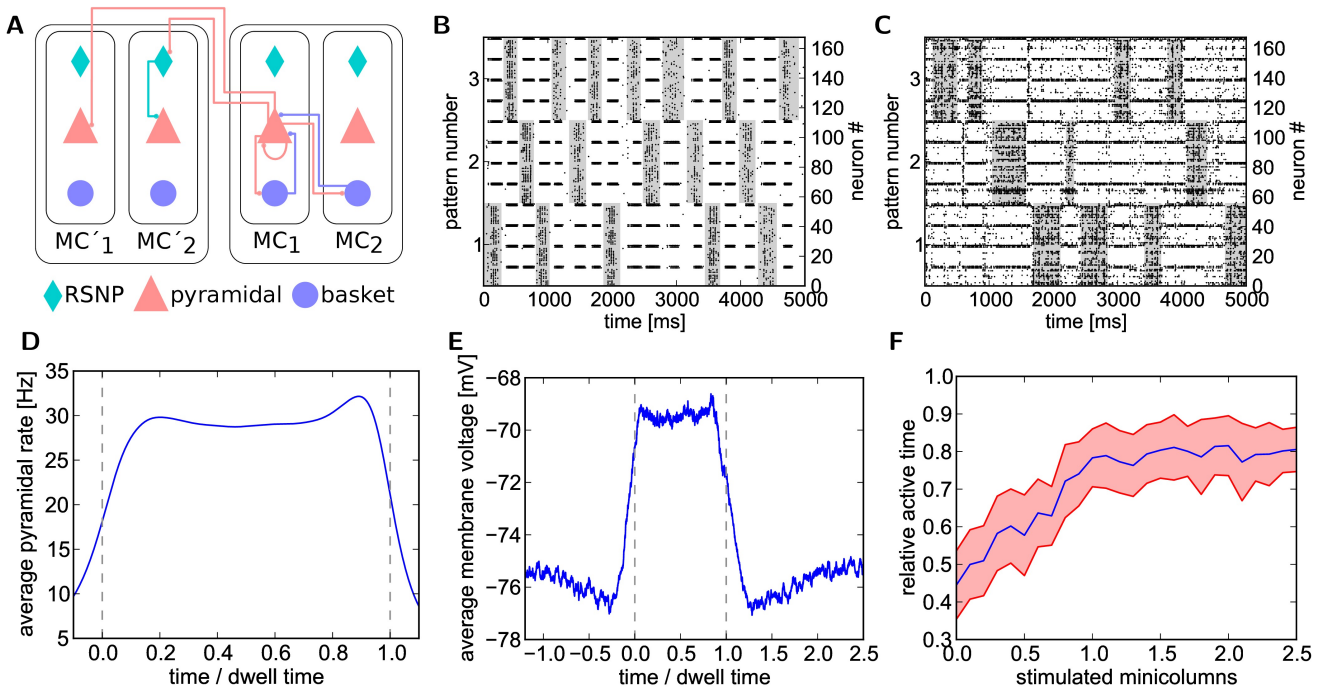

FIGURE 7: **(A)** Schematic of the cortical layer 2/3 attractor memory network. Two hypercolumns, each containing two minicolumns, are shown. For better readability, only connections that are active within an active pattern are depicted. See text for details. **(B)** Software simulation of spiking activity in the cortical attractor network model scaled down to 192 neurons (only pyramidal and RSNP cells shown, basket cells spike almost continously). Minicolumns belonging to the same pattern are grouped together. The broad stripes of activity are generated by pyramidal cells in active attractors. The interlaced narrow stripes of activity represent pairs of RSNP cells, which spike when their home minicolumn is inhibited by other active patterns. **(C)** Same as **B**, but on hardware. The raster plot is noisier and the duration of attractors (dwell time) are less stable than in software due to fixed-pattern noise on neuron and synapse circuits. For better readability, active states are underlied in grey in **B** and **C**. **(D)** Average firing rate of pyramidal cells on the *Spikey* chip inside active patterns. To allow averaging over multiple active periods of varying lengths, all attractor dwell times have been normalized to 1. **(E)** Average membrane potential of pyramidal cells on the *Spikey* chip inside and outside active patterns. **(F)** Pattern completion on the *Spikey* chip. Average values (from multiple runs) depicted in blue, with the standard deviation shown in red. From a relatively equilibrated state where all patterns take turns in being active, additional stimulation (see text) of only a subset of neurons from a given attractor activates the full pattern and enables it to dominate over the other two. The pattern does not remain active indefinitely due to short-term depression in excitatory synapses, thereby still allowing short occasional activations of the other two patterns.

RSNP cells governs the competition among so-called *patterns*, as explained in the following.

In the original model described in Lundqvist et al. (2010), each hypercolumn contains 9 minicolumns, each of which consists of 30 pyramidal, 2 RSNP and 1 basket cells. Within a minicolumn, the pyramidal cells are interconnected and also project onto the 8 closest basket cells within the same hypercolumn. In turn, pyramidal cells in a minicolumn receive projections from all basket cells within the same hypercolumn. All pyramidal cells receive two types of additional excitatory input: an evenly distributed amount of diffuse Poisson noise and specific activation from the cortical layer 4. Therefore, the minicolumns (i.e., the pyramidal populations within) compete among each other in WTA-like fashion, with the winner being determined by the overall strength of the received input.

A pattern (or attractor) is defined as containing exactly one minicolumn from each hypercolumn. Considering only orthogonal patterns (each minicolumn may only belong to a single pattern) and given that all hypercolumns contain an equal amount of minicolumns, the number of patterns in the network is equal to the number of minicolumns per hypercolumn. Pyramidal cells within each minicolumn project onto the pyramidal cells of all the other minicolumns in the same pattern. These connections ensure a spread of local activity throughout the entire pattern. Additionally, the pyramidal cells also

project onto the RSNP cells of all minicolumns belonging to different attractors, which in turn inhibit the pyramidal cells within their minicolumn. This long-range competition enables the winning pattern to completely shut down the activity of all other patterns.

Two additional mechanisms weaken active patterns, thereby facilitating switches between patterns. The pyramidal cells contain an adaptation mechanism which decreases their excitability with every emitted spike. Additionally, the synapses between pyramidal cells are modeled as short-term depressing.

3.4.2. Hardware Emulation

When scaling down the original model (2673 neurons) to the maximum size available on the *Spikey* chip (192 neurons, see Figure 7B for software simulation results), we made use of the essential observation that the number of pyramidal cells can simply be reduced without compensating for it by increasing the corresponding projection probabilities. Also, for less than 8 minicolumns per hypercolumn, all basket cells within a hypercolumn have identical afferent and efferent connectivity patterns, therefore allowing to treat them as a single population. Their total number was decreased, while increasing their efferent projection probabilities accordingly. In general (i.e., except for pyramidal cells), when number and/or size of populations were changed, projection probabilities were scaled in such a way that the total fan-in for each neuron was kept at a constant average. When the maximum fan-in was reached (one afferent synapse for every neuron in the receptive field), the corresponding synaptic weights were scaled up by the remaining factor.

Because neuron and synapse models on the *Spikey* chip are different to the ones used in the original model, we have performed a heuristic fit in order to approximately reproduce the target firing patterns. Neuron and synapse parameters were first fitted in such a way as to generate clearly discernible attractors with relatively high average firing rates (see Figure 7D). Additional tuning was needed to compensate for missing neuronal adaptation, limitations in hardware configurability, parameter ranges and fixed-pattern noise affecting hardware parameters.

During hardware emulations, apart from the appearance of spontaneous attractors given only diffuse Poisson stimulation of the network (Figure 7C), we were able to observe two further interesting phenomena which are characteristic for the original attractor model.

When an attractor becomes active, its pyramidal cells enter a so-called UP state which is characterized by an elevated average membrane potential. Figure 7E clearly shows the emergence of such UP-states on hardware. The onset of an attractor is characterized by a steep rise in pyramidal cell average membrane voltage, which then decays towards the end of the attractor due to synaptic short-term depression and/or competition from other attractors temporarily receiving stronger stimulation. On both flanks of an UP state, the average membrane voltage shows a slight undershoot, due to the inhibition by other active attractors.

A second important characteristic of cortical attractor models is their capability of performing *pattern completion* (Lundqvist et al., 2006). This means that a full pattern can be activated by stimulating only a subset of its constituent pyramidal cells (in the original model, by cells from cortical Layer 4, modeled by us as additional Poisson sources). The appearance of this phenomenon is similar to a phase transition from a resting state to a collective pyramidal UP-state occurring when a critical amount of pyramidal cells are stimulated. To demonstrate pattern completion, we have used the same setup as in the previous experiments, except for one pattern receiving additional stimulation. From an initial equilibrium between the three attractors (approximately equal active time), we have observed the expected sharp transition to a state where the stimulated attractor dominates the other two, occurring when one of its four minicolumns received L4 stimulus (Figure 7F).

The implementation of the attractor memory model is a particularly comprehensive showcase of the configurability and functionality of our neuromorphic platform due to the complexity of both model specifications and emergent dynamics. Starting from these results, the next-generation hardware (Schemmel et al., 2010) will be able to much more accurately model biological behavior, thanks to a more flexible, adapting neuron model and a significantly increased network size.

3.5. INSECT ANTENNAL LOBE MODEL

The high acceleration factor of the *Spikey* chip makes it an attractive platform for neuromorphic data processing. Preprocessing of multivariate data is a common problem in signal and data analysis. In conventional computing, reduction of correlation between input channels is often the first step in the analysis of multidimensional data, achieved, e.g., by *principal component analysis* (PCA). The architecture of the olfactory system maps particularly well onto this problem (Schmuker & Schneider, 2007). We have implemented a network that is inspired by processing principles that have been described in the insect antennal lobe (AL), the first relay station from olfactory sensory neurons to higher brain areas. The function of the AL has been described to decorrelate the inputs from sensory neurons, potentially enabling more efficient memory formation and retrieval (Stopfer et al., 1997; Linster & Smith, 1997; Perez-Orive et al., 2004; Wilson & Laurent, 2005). The mammalian analog of the

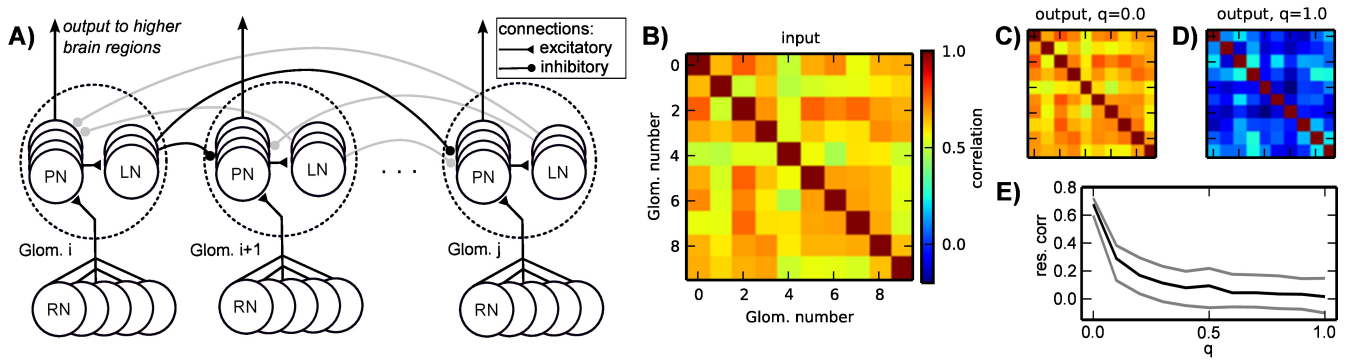

FIGURE 8: (A) Schematic of the insect antennal lobe network. Neuron populations are grouped in glomeruli (outlined by dotted lines), which exert lateral inhibition onto each other. RNs: receptor neurons (input), PNs: projection neurons (output), LNs: inhibitory local neurons. Some connections are grayed out to emphasize the connection principle. **(B)** Correlation matrix of the input data. **(C)** Correlation matrix of the output spike rates (PNs) without lateral inhibition, $q = 0.0$. **(D)** Correlation of the output with homogeneous lateral inhibition, $q = 1.0$. **(E)** Average pairwise correlation between glomeruli (median $\pm$ 20^{th} (black) and 80^{th} (gray) percentile) in dependence of the overall strength of lateral inhibition q.

AL (the olfactory bulb) has been the target of a recent neuromorphic modeling study (Imam et al., 2012b).

The availability of a network building block that achieves channel decorrelation is an important step toward high-performance neurocomputing. The aim of this experiment is to demonstrate that the previously studied rate-based AL model (Schmuker & Schneider, 2007) that reduces rate correlation between input channels is applicable to a spiking neuromorphic hardware system.

3.5.1. Network Topology

In the insect olfactory system, odors are first encoded into neuronal signals by receptor neurons (RNs) which are located on the antenna. RNs send their axons to the AL (Figure 8A). The AL is composed of glomeruli, spherical compartments where RNs project onto local inhibitory neurons (LNs) and projection neurons (PNs). LNs project onto other glomeruli, effecting lateral inhibition. PNs relay the information to higher brain areas where multimodal integration and memory formation takes place.

The architecture of our model reflects the neuronal connectivity in the insect AL (Figure 8A). RNs are modeled as spike train generators, which project onto the PNs in the corresponding glomerulus. The PNs project onto the LNs, which send inhibitory projections to the PNs in other glomeruli.

In biology, the AL network reduces the rate correlation between glomeruli, in order to improve stimulus separability and thus odor identification. Another effect of decorrelation is that the rate patterns encoding the stimuli become sparser, and use the available coding space more efficiently as redundancy is reduced. Our goal was to demonstrate the reduction of rate correlations across glomeruli (*channel correlation*) by the AL-inspired spiking network. To this end, we generated patterns of firing rates with channel correlation. We created a surrogate data set exhibiting channel correlation using a copula, a technique that allows to generate correlated series of samples from an arbitrary random distribution and a covariance matrix (Nelsen, 1998). The covariance matrix was uniformly set to a target correlation of 0.6. Using this copula, we sampled 100 ten-dimensional data vectors from an exponential distribution. In the biological context, this is equivalent to having a repertoire of 100 odors, each encoded by ten receptors, and the firing rate of each input channel following a decaying exponential distribution. Values larger than e were clipped and the distribution was mapped to the interval $[0, 1]$ by applying $v = v/e$ for each value v. These values were then converted into firing rates between 20 and 55 spikes/s. The ten-dimensional data vector was presented to the network by mapping the ten firing rates onto the ten glomeruli, setting all single RNs in each glomerulus to fire at the respective target rates. Rates were converted to spike trains individually for each RN using the Gamma process with $\gamma = 5$. Each data vector was presented to the network for the duration of one second by making the RNs of each glomerulus fire with the specified rate. The inhibitory weights between glomeruli were uniform, i.e., all inhibitory connections shared the same weight. During one second of stimulus presentation, output rates were measured from PNs. One output rate per glomerulus was obtained by averaging the firing rate of all PNs in a glomerulus.

We have used 6 RN input streams per glomerulus, projecting in an all-to-all fashion onto 7 PNs, which in turn projected on 3 LNs per glomerulus.

3.5.2. Hardware Emulation

The purpose of the presented network was to reduce rate correlation between input channels. As in other models, fixed-pattern noise across neurons had a detrimental effect on the function of the network. We exploited the specific structure of our network to implement more efficient calibration than can be provided by standard calibration methods (Section 2.4). Our calibration algorithm targeted PNs and LNs in the first layer of the network. During calibration, we turned off all projections between glomeruli. Its aim was to achieve a homogeneous response across PNs and LNs respectively, i.e., within $\pm$ 10% of a target rate. The target rate was chosen from the median response rate of uncalibrated neurons. For neurons whose response rate was too high it was sufficient to reduce the synaptic weight of the excitatory input from RNs. For those neurons with a too low rate the input strength had to be increased. The excitatory synaptic weight of the input from RNs was initially already at its maximum value and could not be increased. As a workaround we used PNs from the same glomerulus to add additional excitatory input to those "weak" neurons. We ensured that no recurrent excitatory loops were introduced by this procedure. If all neurons in a glomerulus were too weak, we recruit another external input stream to achieve the desired target rate. Once the PNs were successfully calibrated (less than 10% deviation from the target rate), we used the same approach to calibrate the LNs in each glomerulus.

To assess the performance of the network we have compared the channel correlation in the input and in the output. The channel correlation matrix $\mathbf{C}$ was computed according to

$$C_{i,j} = d^{\text{Pearson}}(\boldsymbol{\nu}_{\text{glom}.i}, \boldsymbol{\nu}_{\text{glom}.j}) \,, \tag{5}$$

with $d^{\text{Pearson}}(\cdot,\cdot)$ the Pearson correlation coefficient between two vectors. For the input correlation matrix $\mathbf{C}^{\text{input}}$, the vector $\boldsymbol{\nu}_{\text{glom}.i}$ contained the average firing rates of the six RNs projecting to the ith glomerulus, with each element of this vector for one stimulus presentation. For the output correlation matrix $\mathbf{C}^{\text{output}}$ we used the rates from PNs instead of RNs. Thus, we obtained 10×10 matrices containing the rate correlations for each pair of input or output channels.

Figure 8B depicts the correlation matrix $\mathbf{C}^{\text{input}}$ for the input firing rates. When no lateral inhibition is present, $\mathbf{C}^{\text{input}}$ matches $\mathbf{C}^{\text{output}}$ (Figure 8C). We have systematically varied the strength of lateral inhibition by scaling all inhibitory weights by a factor q, with $q = 0$ for zero lateral inhibition and $q = 1$ for inhibition set to its maximal strength. With increasing lateral inhibition, off-diagonal values in $\mathbf{C}^{\text{output}}$ approach zero and output channel correlation is virtually gone (Figure 8D). The amount of residual correlation to be present in the output can be controlled by adjusting the strength of lateral inhibition (Figure 8E).

Taken together, we demonstrated the implementation of an olfaction-inspired network to remove correlation between input channels on the *Spikey* chip. This network can serve as a preprocessing module for data analysis applications to be implemented on the *Spikey* chip. An interesting candidate for such an application is a spiking network for supervised classification, which may benefit strongly from reduced channel correlations for faster learning and better discrimination (Häusler et al., 2011).

3.6. LIQUID STATE MACHINE

Liquid state machines (LSMs) as proposed by Maass et al. (2002) and Jaeger (2001) provide a generic framework for computation on continuous input streams. The *liquid*, a recurrent network, projects an input into a high-dimensional space which is subsequently read out. It has been proven that LSMs have universal computational power for computations with fading memory on functions of time (Maass et al., 2002). In the following, we show that classification performance of a LSM emulated on our hardware is comparable to the corresponding computer simulation. Synaptic weights of the readout are iteratively learned on-chip, which inherently compensates for fixed-pattern noise. A trained system can then be used as an autonomous and very fast spiking classifier.

3.6.1. Network Topology

The LSM consists of two major components: the recurrent liquid network itself and a spike-based classifier (Figure 9A). A general purpose liquid needs to meet the separation property (Maass et al., 2002), which requires that different inputs are mapped to different outputs, for a wide range of possible inputs. Therefore, we use a network topology similar to the one proposed by Bill et al. (2010). It consists of an excitatory and inhibitory population with a ratio of 80:20 excitatory to inhibitory neurons. Both populations had recurrent as well as feedforward connections. Each neuron in the liquid receives 4 inputs from the 32 excitatory and 32 inhibitory sources, respectively. All other connection probabilities are illustrated in Figure 9.

The readout is realized by means of a tempotron (Gütig & Sompolinsky, 2006), which is compatible with our hardware due to its spike-based nature. Furthermore, its modest single neuron implementation leaves most hardware resources to the liquid. The afferent synaptic weights are trained with the method described in Gütig & Sompolinsky (2006), which effectively implements gradient descent dynamics. Upon training, the tempotron distinguishes

between two input classes by emitting either one or no spike within a certain time window. The former is artificially enforced by blocking all further incoming spikes after the first spike occurrence.

The PSP kernel of a LIF neuron with current-based synapses is given by

$$K(t-t_i) = A\left(e^{-\frac{t-t_i}{\tau_\mathrm{m}}} - e^{-\frac{t-t_i}{\tau_\mathrm{s}}}\right)\cdot\Theta(t-t_i)\ , \quad (6)$$

with the membrane time constant τ_m and the synaptic time constant τ_s, respectively. Here, A denotes a constant PSP scaling factor, t_i the time of the ith incoming spike and $\Theta(t)$ the Heaviside step function.

During learning, weights are updated as follows

$$\Delta w_j^n = \begin{cases} 0 & \text{correct} \\ \alpha(n)\sum_{t_{i,j}<t_\mathrm{max}} K(t_\mathrm{max}-t_{i,j}) & \text{erroneous,} \end{cases} \quad (7)$$

where Δw_j^n is the weight update corresponding to the jth afferent neuron after the nth learning iteration with learning rate $\alpha(n)$. The spike time of the tempotron, or otherwise the time of highest membrane potential, is denoted with t_max. In other words, for trials where an erroneous spike was elicited, the excitatory afferents with a causal contribution to this spike are weakened and inhibitory ones are strengthened according to Equation 7. In case the tempotron did not spike even though it should have, the weights are modulated the other way round, i.e. excitatory weights are strengthened and inhibitory ones are weakened. This learning rule has been implemented on hardware with small modifications, due to the conductance-based nature of the hardware synapses (see below).

The tempotron is a binary classifier, hence any task needs to be mapped to a set of binary decisions. Here, we have chosen a simple binary task adapted from Maass et al. (2002), to evaluate the performance of the LSM. The challenge was to distinguish spike train segments in a continuous data stream composed of two templates with identical rates (denoted X and Y in Figure 9A). In order to generate the input, we cut the template spike trains into segments of 50 ms duration. We then composed the spike sequence to be presented to the network by randomly picking a spike segment from either X or Y in each time window (see Figure 9 for a schematic). Additionally, we added spike timing jitter from a normal distribution with a standard deviation of $\sigma = 1$ ms to each spike. For each experiment run both for training and classification, the composed spike sequence was then streamed into the liquid. Tempotrons were given the liquid activity as input and trained to identify whether the segment within the previous time window originated from sequence X or Y. In a second attempt, we trained the tempotron to identify the origin of the pattern presented in the window at -100 to -150 ms (that is, the second to the last window). Not only did this task allow to determine the classification capabilities of the LSM, but it also put the liquid's fading memory to the test: classification of a segment further back in time becomes increasingly difficult.

3.6.2. Hardware Emulation

The liquid itself does not impose any strong requirements on the hardware since virtually any network is suitable as long as the separation property is satisfied. We adapted a network from Bill et al. (2010) which, in a similar form, had already been implemented on our hardware. However, STP was disabled, because at the time of the experiment it was not possible to exclusively enable STP for the liquid without severely affecting the performance of the tempotron.

The hardware implementation of the tempotron required more attention, since only conductance-based synapses are available. The dependence of spike efficacies on the actual membrane potential was neglected, because the rest potential was chosen to be close to the firing threshold, with the reversal potentials far away. However, the asymmetric distance of excitatory and inhibitory reversal potentials from the sub-threshold regime needed compensation. This was achieved by scaling all excitatory weights by $(\overline{V_\mathrm{m}} - E_\mathrm{inh})/(\overline{V_\mathrm{m}} - E_\mathrm{exc})$, where $\overline{V_\mathrm{m}}$ corresponds to the mean neuron membrane voltage and $E_\mathrm{exc}/E_\mathrm{inh}$ is the excitatory/inhibitory reversal potentials. Discontinuities in spike efficacies for synapses changing from excitatory to inhibitory or vice versa were avoided by prohibiting such transitions. Finally, membrane potential shunting after the first spike occurrence is neither possible on our hardware nor very biological and had therefore been neglected, as already proposed by Gütig & Sompolinsky (2006).

Even though the tempotron was robust against fixed-pattern noise due to on-chip learning, the liquid required modifications. Therefore, firing thresholds were tuned independently in software and hardware to optimize the memory capacity and avoid violations of the separation property. Since hardware neurons share firing thresholds, the tempotron was affected accordingly (see Table 1). Additionally, the learning curve $\alpha(n)$ was chosen individually for software and hardware due to the limited resolution of synaptic weights on the latter.

The results for software and hardware implementations are illustrated in Figure 9B. Both LSMs performed at around 90% classification correctness for the spiketrain segment that lied 50 ms to 100 ms in the past with respect to the end of the stimulus. For inputs lying even further away in time, performances dropped to chance level (50%

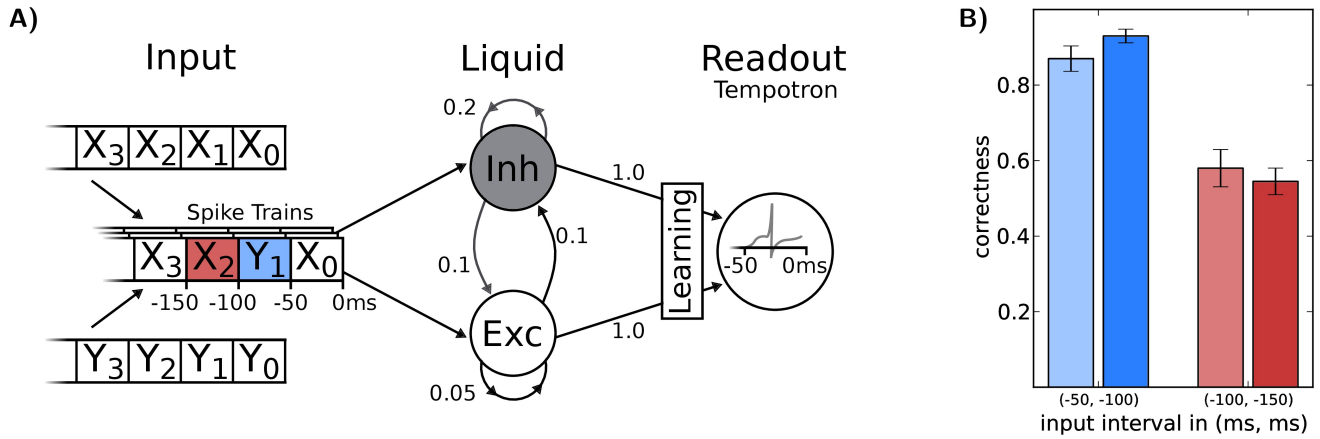

FIGURE 9: (A) Schematic of the LSM and the given task. Spike sources are composed of $50\,\mathrm{ms}$ segments drawn from two template spike trains (X and Y). These patterns are streamed into the liquid (with descending index), which is a network consisting of 191 neurons, leaving one neuron for the tempotron. Connection probabilities are depicted next to each connection (arrows). The tempotron is trained to classify the origin (X or Y) of the spike train segment in the previous $50\,\mathrm{ms}$ time window (with index 1). **(B)** The classification performance of the LSM measured over 200 samples after 1000 training iterations for both hardware (lighter) and software (darker) implementation.

for a binary task), independent of the simulation backend.

Regarding the classification capabilities of the LSM, our current implementation allows a large variety of tasks to be performed. Currently, e.g., we are working on handwritten digit recognition with the very same setup on the *Spikey* chip. Even without a liquid, our implementation of the tempotron (or populations thereof) makes an excellent neuromorphic classifier, given its bandwidth-friendly sparse response and robustness against fixed-pattern noise.

4. DISCUSSION

We have successfully implemented a variety of neural microcircuits on a single universal neuromorphic substrate, which is described in detail by Schemmel et al. (2006). All networks show activity patterns qualitatively and to some extent also quantitatively similar to those obtained by software simulations. The corresponding reference models found in literature have not been modified significantly and network topologies have been identical for hardware emulation and software simulation, if not stated otherwise. In particular, the emulations benefit from the advantages of our neuromorphic implementation, namely inherent parallelism and accelerated operation compared to software simulations on conventional von-Neumann machines. Previous accounts of networks implemented on the *Spikey* system include computing with high-conductance states (Kaplan et al., 2009), self-stabilizing recurrent networks (Bill et al., 2010), and simple emulations of cortical layer 2/3 attractor networks (Brüderle et al., 2011).

In this contribution, we have presented a number of new networks and extensions of previous implementations. Our synfire chain implementation achieves reliable signal propagation over years of biological time from one single stimulation, while synchronizing and filtering these signals (Section 3.1). Our extension of the network from Bill et al. (2010) to exhibit asynchronous irregular firing behavior is an important achievement in the context of reproducing stochastic activity patterns found in cortex (Section 3.2). We have realized soft winner-take-all networks on our hardware system (Section 3.3), which are essential building blocks for many cortical models involving some kind of attractor states (e.g., the decision-making model by Soltani & Wang, 2010). The emulated cortical attractor model provides an implementation of working memory for computation with cortical columns (Section 3.4). Additionally, we have used the *Spikey* system for preprocessing of multivariate data inspired by biological archetypes (Section 3.5) and machine learning (Section 3.6). Most of these networks allocate the full number of neurons receiving input from one synapse array on the *Spikey* chip, but with different sets of neuron and synapse parameters and especially vastly different connectivity patterns, thereby emphasizing the remarkable configurability of our neuromorphic substrate.

However, the translation of such models requires modifications to allow execution on our hardware. The most prominent cause for such modifications is fixed-pattern noise across analog hardware neurons and synapses. In most cases, especially when population rate coding is involved, it is sufficient to compensate for this variability

by averaging spiking activity over many neurons. For the data decorrelation and machine learning models, we have additionally trained the synaptic weights on the chip to achieve finer equilibration of the variability at critical network nodes. Especially when massive downscaling is required in order for models to fit onto the substrate, fixed pattern noise presents an additional challenge because the same amount of information needs to be encoded by fewer units. For this reason, the implementation of the cortical attractor memory network required additional heuristic activity fitting procedures.

The usability of the *Spikey* system, especially for neuroscientists with no neuromorphic engineering background, is provided by an integrated development environment. We envision that the configurability made accessible by such a software environment will encourage a broader neuroscience community to use our hardware system. Examples of use would be the acceleration of simulations as well as the investigation of the robustness of network models against parameter variability, both between computational units and between trials, as e.g. published by Brüderle et al. (2010) and Schmuker et al. (2011). The hardware system can be efficiently used without knowledge about the hardware implementation on transistor level. Nevertheless, users have to consider basic hardware constraints, as e.g., shared parameters. Networks can be developed using the PyNN metalanguage and optionally be prototyped on software simulators before running on the *Spikey* system (Davison et al., 2009; Brüderle & Müller et al., 2009). This rather easy configuration and operation of the *Spikey* chip allows the implementation of many other neural network models.

There exist also boundaries to the universal applicability of our hardware system. One limitation inherent to this type of neuromorphic device is the choice of implemented models for neuron and synapse dynamics. Models requiring, e.g., neuronal adaptation or exotic synaptic plasticity rules are difficult, if not impossible to be emulated on this substrate. Also, the total number of neurons and synapses set a hard upper bound on the size of networks that can be emulated. However, the next generation of our highly accelerated hardware system will increase the number of available neurons and synapses by a factor of 10^3, and provide extended configurability for each of these units (Schemmel et al., 2010).

The main purpose of our hardware system is to provide a flexible platform for highly accelerated emulation of spiking neuronal networks. Other research groups pursue different design goals for their hardware systems. Some focus on dedicated hardware providing specific network topologies (e.g., Merolla & Boahen, 2006; Chicca et al., 2007), or comprising few neurons with more complex dynamics (e.g., Chen et al., 2010; Grassia et al., 2011; Brink et al., 2012). Others develop hardware systems of comparable configurability, but operate in biological real-time, mostly using off-chip communication (Vogelstein et al., 2007; Choudhary et al., 2012). Purely digital systems (Merolla et al., 2011; Furber et al., 2012; Imam et al., 2012a) and field-programmable analog arrays (FPAA; Basu et al., 2010) provide even more flexibility in configuration than our system, but have much smaller acceleration factors.

With the ultimate goal of brain size emulations, there exists a clear requirement for increasing the size and complexity of neuromorphic substrates. An accompanying upscaling of the fitting and calibration procedures presented here appears impractical for such orders of magnitude and can only be done for a small subset of components. Rather, it will be essential to step beyond simulation equivalence as a quality criterion for neuromorphic computing, and to develop a theoretical framework for circuits that are robust against, or even exploit the inherent imperfections of the substrate for achieving the required computational functions.

ACKNOWLEDGMENTS

We would like to thank Dan Husmann, Stefan Philipp, Bernhard Kaplan, and Moritz Schilling for their essential contributions to the neuromorphic platform, Johannes Bill, Jens Kremkow, Anders Lansner, Mikael Lundqvist and Emre Neftci for assisting with the hardware implementation of the network models, Oliver Breitwieser for data analysis and Venelin Petkov for characterisation measurements of the *Spikey* chip.

The research leading to these results has received funding by the European Union 6th and 7th Framework Programme under grant agreement no. 15879 (FACETS), no. 269921 (BrainScaleS) and no. 243914 (Brain-i-Nets). Michael Schmuker has received support from the german ministry for research and education (BMBF) to Bernstein Center for Computational Neuroscience Berlin (grant no. 01GQ1001D) and from Deutsche Forschungsgemeinschaft within SPP 1392 (grant no. SCHM 2474/1-1).

The main contributors for each section are denoted with author initials: neuromorphic system (AG & EM); synfire chain (PM); balanced random network (TP); soft winner-take-all network (TP); cortical attractor model (MP); antennal lobe model (MS); liquid state machine (SJ). All authors contributed to writing this paper.

REFERENCES

Abeles, M., Hayon, G., & Lehmann, D. (2004). Modeling compositionality by dynamic binding of synfire chains. *J. Comput. Neurosci. 17*(2), 179–201.

Aertsen, A., Diesmann, M., & Gewaltig, M.-O. (1996). Propagation of synchronous spiking activity in feedforward neural networks. *J. Physiol. (Paris) 90*(3/4), 243–247.

Akay, M. (2007). *Handbook of neural engineering - Part II: Neuro-nanotechnology: Artificial implants and neural prosthesis.* Wiley-IEEE Press.

Aviel, Y., Mehring, C., Abeles, M., & Horn, D. (2003). On embedding synfire chains in a balanced network. *Neural Comput. 15*(6), 1321–1340.

Baker, S. N., Spinks, R., Jackson, A., & Lemon, R. N. (2001). Synchronization in monkey motor cortex during a precision grip task. I. Task-dependent modulation in single-unit synchrony. *J. Neurophysiol. 85*(2), 869–85.

Basu, A., Ramakrishnan, S., Petre, C., Koziol, S., Brink, S., & Hasler, P. (2010). Neural dynamics in reconfigurable silicon. *IEEE Trans. Biomed. Circuits Syst. 4*(5), 311–319.

Bi, G., & Poo, M. (1998). Synaptic modifications in cultured hippocampal neurons: Dependence on spike timing, synaptic strength, and postsynaptic cell type. *J. Neurosci. 18*, 10464–10472.

Bill, J., Schuch, K., Brüderle, D., Schemmel, J., Maass, W., & Meier, K. (2010). Compensating inhomogeneities of neuromorphic VLSI devices via short-term synaptic plasticity. *Front. Comput. Neurosci. 4*(129).

Brink, S., Nease, S., Hasler, P., Ramakrishnan, S., Wunderlich, R., Basu, A., & Degnan, B. (2012). A learning-enabled neuron array IC based upon transistor channel models of biological phenomena. *IEEE Trans. Biomed. Circuits Syst. PP*(99), 1.

Brüderle, D., Bill, J., Kaplan, B., Kremkow, J., Meier, K., Müller, E., & Schemmel, J. (2010). Simulator-like exploration of cortical network architectures with a mixed-signal VLSI system. In *Proceedings of the 2010 International Symposium on Circuits and Systems (ISCAS)*. IEEE Press.

Brüderle, D., Müller, E., Davison, A., Muller, E., Schemmel, J., & Meier, K. (2009). Establishing a novel modeling tool: A python-based interface for a neuromorphic hardware system. *Fed. Proc. 3*(17).

Brüderle, D., Petrovici, M., Vogginger, B., Ehrlich, M., Pfeil, T., Millner, S., Grübl, A., Wendt, K., Müller, E., Schwartz, M.-O., Husmann de Oliveira, D., Jeltsch, S., Fieres, J., Schilling, M., Müller, P., Breitwieser, O., Petkov, V., Muller, L., Davison, A. P., Krishnamurthy, P., Kremkow, J., Lundqvist, M., Muller, E., Partzsch, J., Scholze, S., Zühl, L., Destexhe, A., Diesmann, M., Potjans, T. C., Lansner, A., Schüffny, R., Schemmel, J., & Meier, K. (2011). A comprehensive workflow for general-purpose neural modeling with highly configurable neuromorphic hardware systems. *Biol. Cybern. 104*, 263–296.

Brunel, N. (2000). Dynamics of sparsely connected networks of excitatory and inhibitory spiking neurons. *J. Comput. Neurosci. 8*(3), 183–208.

Carnevale, T., & Hines, M. (2006). *The NEURON Book*. Cambridge: Cambridge University Press.

Chen, H., Saïghi, S., Buhry, L., & Renaud, S. (2010). Real-time simulation of biologically realistic stochastic neurons in VLSI. *IEEE Trans. Neural Netw. 21*(9), 1511–1517.

Chicca, E., Whatley, A. M., Lichtsteiner, P., Dante, V., Delbruck, T., Del Giudice, P., Douglas, R. J., & Indiveri, G. (2007). A multichip pulse-based neuromorphic infrastructure and its application to a model of orientation selectivity. *IEEE Trans. Circuits Syst. I, Reg. Papers 54*(5), 981–993.

Choudhary, S., Sloan, S., Fok, S., Neckar, A., Trautmann, E., Gao, P., Stewart, T., Eliasmith, C., & Boahen, K. (2012). Silicon neurons that compute. In A. Villa, W. Duch, P. Érdi, F. Masulli, & G. Palm (Eds.), *Artificial Neural Networks and Machine Learning – ICANN 2012*, Volume 7552 of *Lecture Notes in Computer Science*, pp. 121–128. Springer Berlin / Heidelberg.

Davison, A., Brüderle, D., Eppler, J. M., Kremkow, J., Muller, E., Pecevski, D., Perrinet, L., & Yger, P. (2009). PyNN: a common interface for neuronal network simulators. *Front. Neuroinformatics 2*(11).

Davison, A., Muller, E., Brüderle, D., & Kremkow, J. (2010). A common language for neuronal networks in software and hardware. *The Neuromorphic Engineer*.

Dayan, P., & Abbott, L. F. (2001). *Theoretical Neuroscience*. Cambridge: MIT Press.

Diesmann, M., Gewaltig, M.-O., & Aertsen, A. (1999). Stable propagation of synchronous spiking in cortical neural networks. *Nature 402*(6761), 529–533.

Douglas, R. J., & Martin, K. A. C. (2004). Neuronal circuits of the neocortex. *Annu. Rev. Neurosci. 27*, 419–451.

Eppler, J. M., Helias, M., Muller, E., Diesmann, M., & Gewaltig, M. (2009). PyNEST: a convenient interface to the NEST simulator. *Front. Neuroinformatics 2*, 12.

Farquhar, E., & Hasler, P. (2005). A bio-physically inspired silicon neuron. *IEEE Trans. Circuits Syst. I, Reg. Papers 52*(3), 477–488.

Fieres, J. and. Grübl, A., Philipp, S., Meier, K., Schemmel, J., & Schürmann, F. (2004). A platform for parallel operation of VLSI neural networks. In *Proceedings of the 2004 Brain Inspired Cognitive Systems Conference (BICS)*, University of Stirling, Scotland, UK.

Furber, S., Lester, D., Plana, L., Garside, J., Painkras, E., Temple, S., & Brown, A. (2012). Overview of the spinnaker system architecture. *IEEE Trans. Comp. PP*(99), 1.

Gao, P., Benjamin, B., & Boahen, K. (2012). Dynamical system guided mapping of quantitative neuronal models onto neuromorphic hardware. *IEEE Trans. Circuits Syst. I, Reg. Papers 59*(10), 2383–2394.

Gewaltig, M.-O., & Diesmann, M. (2007). NEST (NEural Simulation Tool). *Scholarpedia 2*(4), 1430.

Grassia, F., Buhry, L., Lévi, T., Tomas, J., Destexhe, A., & Saïghi, S. (2011). Tunable neuromimetic integrated system for emulating cortical neuron models. *Front. Neurosci. 5*(134).

Grossberg, S. (1973). Contour enhancement, short term memory, and constancies in reverberating neural networks. *Stud. Appl. Math. 52*(3), 213–257.

Gupta, A., Wang, Y., & Markram, H. (2000). Organizing principles for a diversity of GABAergic interneurons and synapses in the neocortex. *Science 287*, 273–278.

Gütig, R., & Sompolinsky, H. (2006). The tempotron: a neuron that learns spike timing-based decisions. *Nat. Neurosci. 9*(3), 420–428.

Häusler, C., Nawrot, M. P., & Schmuker, M. (2011). A spiking neuron classifier network with a deep architecture inspired by the olfactory system of the honeybee. In *2011 5th International IEEE/EMBS Conference on Neural Engineering*, pp. 198–202. IEEE Press.

Hines, M., Davison, A. P., & Muller, E. (2009). Neuron and python. *Front. Neuroinformatics 3*.

Imam, N., Akopyan, F., Arthur, J., Merolla, P., Manohar, R., & Modha, D. (2012a). A digital neurosynaptic core using event-driven QDI circuits. In *IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC)*, pp. 25–32.

Imam, N., Cleland, T. A., Manohar, R., Merolla, P. A., Arthur, J. V., Akopyan, F., & Modha, D. S. (2012b). Implementation of olfactory bulb glomerular layer computations in a digital neurosynaptic core. *Front. Neurosci. 6*(83).

Indiveri, G., Chicca, E., & Douglas, R. (2006). A VLSI array of low-power spiking neurons and bistable synapses with spike-timing dependent plasticity. *IEEE Trans. Neural Netw. 17*(1), 211–221.

Indiveri, G., Chicca, E., & Douglas, R. (2009). Artificial cognitive systems: From VLSI networks of spiking neurons to neuromorphic cognition. *Cognitive Computation 1*(2), 119–127.

Indiveri, G., Linares-Barranco, B., Hamilton, T. J., van Schaik, A., Etienne-Cummings, R., Delbruck, T., Liu, S.-C., Dudek, P., Häfliger, P., Renaud, S., Schemmel, J., Cauwenberghs, G., Arthur, J., Hynna, K., Folowosele, F., Saighi, S., Serrano-Gotarredona, T., Wijekoon, J., Wang, Y., & Boahen, K. (2011). Neuromorphic silicon neuron circuits. *Front. Neurosci. 5*(73).

Itti, L., & Koch, C. (2001). Computational modeling of visual attention. *Nat. Rev. Neurosci. 2*(3), 194–203.

Jaeger, H. (2001). The "echo state" approach to analysing and training recurrent neural networks. Technical Report GMD Report 148, German National Research Center for Information Technology.

Kaplan, B., Brüderle, D., Schemmel, J., & Meier, K. (2009). High-conductance states on a neuromorphic hardware system. In *Proceedings of the 2009 International Joint Conference on Neural Networks (IJCNN)*, Atlanta, pp. 1524–1530.

Kremkow, J., Aertsen, A., & Kumar, A. (2010). Gating of signal propagation in spiking neural networks by balanced and correlated excitation and inhibition. *J. Neurosci. 30*(47), 15760–15768.

Kremkow, J., Perrinet, L. U., Masson, G. S., & Aertsen, A. (2010). Functional consequences of correlated excitatory and inhibitory conductances in cortical networks. *J. Comput. Neurosci. 28*(3), 579–594.

Kumar, A., Schrader, S., Aertsen, A., & Rotter, S. (2008). The high-conductance state of cortical networks. *Neural Comput. 20*(1), 1–43.

Lazzaro, J., Ryckebusch, S., Mahowald, M., & Mead, C. (1988). Winner-take-all networks of *o(n)* complexity. In *Advances in Neural Information Processing Systems (NIPS)*, Denver, pp. 703–711.

Linster, C., & Smith, B. H. (1997). A computational model of the response of honey bee antennal lobe circuitry to odor mixtures: overshadowing, blocking and unblocking can arise from lateral inhibition. *Behav. Brain. Res. 87*, 1–14.

Litvak, V., Sompolinsky, H., Segev, I., & Abeles, M. (2003). On the transmission of rate code in long feed-forward networks with excitatory-inhibitory balance. *J. Neurosci. 23*(7), 3006–3015.

Lundqvist, M., Compte, A., & Lansner, A. (2010). Bistable, irregular firing and population oscillations in a modular attractor memory network. *PLoS Comput. Biol. 6*(6), e1000803.

Lundqvist, M., Rehn, M., Djurfeldt, M., & Lansner, A. (2006). Attractor dynamics in a modular network model of neocortex. *Network: Comput. Neural Systems 17*(3), 253–276.

Maass, W. (2000). On the computational power of winner-take-all. *Neural Comput. 12*(11), 2519–2535.

Maass, W., Natschläger, T., & Markram, H. (2002). Real-time computing without stable states: a new framwork for neural compuation based on perturbation. *Neural Comput. 14*(11), 2531–2560.

Marder, E., & Goaillard, J.-M. (2006). Variability, compensation and homeostasis in neuron and network function. *Nat. Rev. Neurosci. 7*(7), 563–574.

Markram, H., Wang, Y., & Tsodyks, M. (1998). Differential signaling via the same axon of neocortical pyramidal neurons. *Proc. Natl. Acad. Sci. USA 95*(9), 5323–5328.

Mead, C. (1989). *Analog VLSI and neural systems*. Boston, MA, USA: Addison-Wesley.

Merolla, P., Arthur, J., Akopyan, F., Imam, N., Manohar, R., & Modha, D. (2011). A digital neurosynaptic core using embedded crossbar memory with 45pJ per spike in 45nm. In *Proceedings of the 2011 Custom Integrated Circuits Conference (CICC)*, pp. 1–4. IEEE Press.

Merolla, P., & Boahen, K. (2006). Dynamic computation in a recurrent network of heterogeneous silicon neurons. In *Proceedings of the 2006 International Symposium on Circuits and Systems (ISCAS)*, Island of Kos, pp. 4539–4542. IEEE Press.

Neftci, E., Chicca, E., Indiveri, G., & Douglas, R. (2011). A systematic method for configuring VLSI networks of spiking neurons. *Neural Comput. 23*(10), 2457–2497.

Neftci, E., & Indiveri, G. (2010). A device mismatch compensation method for VLSI neural networks. In *Proceedings of the 2010 Biomedical Circuits and Systems Conference (BioCAS)*, pp. 262–265. IEEE Press.

Nelsen, R. B. (1998). *An Introduction to Copulas (Lecture Notes in Statistics)* (1 ed.). Springer.

Oster, M., Douglas, R., & Liu, S. (2009). Computation with spikes in a winner-take-all network. *Neural Comput. 21*(9), 2437–2465.

Perez-Orive, J., Bazhenov, M., & Laurent, G. (2004). Intrinsic and circuit properties favor coincidence detection for decoding oscillatory input. *J. Neurosci. 24*(26), 6037–47.

Perkel, D. H., Gerstein, G. L., & Moore, G. P. (1967). Neuronal spike trains and stochastic point processes. II. Simultaneous spike trains. *Biophys. J. 7*(4), 419–440.

Pfeil, T., Potjans, T. C., Schrader, S., Potjans, W., Schemmel, J., Diesmann, M., & Meier, K. (2012). Is a 4-bit synaptic weight resolution enough? - constraints on enabling spike-timing dependent plasticity in neuromorphic hardware. *Front. Neurosci. 6*(90).

Prut, Y., Vaadia, E., Bergman, H., Haalman, I., Hamutal, S., & Abeles, M. (1998). Spatiotemporal structure of cortical activity: Properties and behavioral relevance. *J. Neurophysiol. 79*(6), 2857–2874.

Renaud, S., Tomas, J., Lewis, N., Bornat, Y., Daouzli, A., Rudolph, M., Destexhe, A., & Saïghi, S. (2010). PAX: A mixed hardware/software simulation platform for spiking neural networks. *Neural Networks 23*(7), 905–916.

Rolls, E. T., & Deco, G. (2010). *The Noisy Brain: Stochastic Dynamics as a Principle*. Oxford University Press.

Schemmel, J., Brüderle, D., Grübl, A., Hock, M., Meier, K., & Millner, S. (2010). A wafer-scale neuromorphic hardware system for large-scale neural modeling. In *Proceedings of the 2010 International Symposium on Circuits and Systems (ISCAS)*, Paris, pp. 1947–1950. IEEE Press.

Schemmel, J., Brüderle, D., Meier, K., & Ostendorf, B. (2007). Modeling synaptic plasticity within networks of highly accelerated I&F neurons. In *Proceedings of the 2007 International Symposium on Circuits and Systems (ISCAS)*, New Orleans. IEEE Press.

Schemmel, J., Grübl, A., Meier, K., & Müller, E. (2006). Implementing synaptic plasticity in a VLSI spiking neural network model. In *Proceedings of the 2006 International Joint Conference on Neural Networks (IJCNN)*, Vancouver, pp. 1–6. IEEE Press.

Schmuker, M., Brüderle, D., Schrader, S., & Nawrot, M. P. (2011). Ten thousand times faster: Classifying multidimensional data on a spiking neuromorphic hardware system. *Front. Comput. Neurosci. Conference Abstract: BC11 : Computational Neuroscience & Neurotechnology Bernstein Conference & Neurex Annual Meeting 2011*(109).

Schmuker, M., & Schneider, G. (2007). Processing and classification of chemical data inspired by insect olfaction. *Proc. Natl. Acad. Sci. USA 104*(51), 20285–20289.

Schrader, S., Diesmann, M., & Morrison, A. (2010). A compositionality machine realized by a hierarchic architecture of synfire chains. *Front. Comput. Neurosci. 4*, 154.

Schrader, S., Grün, S., Diesmann, M., & Gerstein, G. (2008). Detecting synfire chain activity using massively parallel spike train recording. *J. Neurophysiol. 100*, 2165–2176.

Serrano-Gotarredona, R., Oster, M., Lichtsteiner, P., Linares-Barranco, A., Paz-Vicente, R., Gómez-Rodríguez, F., Kolle Riis, H., Delbrück, T., Liu, S.-C., Zahnd, S., Whatley, A. M., Douglas, R., Häfliger, P., Jimenez-Moreno, G., Civit, A., Serrano-Gotarredona, T., Acosta-Jiménez, A., & Linares-Barranco, B. (2006). AER building blocks for multi-layer multi-chip neuromorphic vision systems. In Y. Weiss, B. Schölkopf, & J. Platt (Eds.), *anips*, pp. 1217–1224. Cambridge, MA: MIT Press.

Soltani, A., & Wang, X.-J. (2010). Synaptic computation underlying probabilistic inference. *Nat. Neurosci. 13*(1), 112–9.

Song, S., Miller, K. D., & Abbott, L. F. (2000). Competitive Hebbian learning through spike-timing-dependent synaptic plasticity. *Nat. Neurosci. 3*(9), 919–926.

Stopfer, M., Bhagavan, S., Smith, B. H., & Laurent, G. (1997). Impaired odour discrimination on desynchronization of odour-encoding neural assemblies. *Nature 390*(6655), 70–4.

Tetzlaff, T., Morrison, A., Timme, M., & Diesmann, M. (2005). Heterogeneity breaks global synchrony in large networks. In *Proceedings of the 30th Göttingen Neurobiology Conference*.

Tsodyks, M. V., & Markram, H. (1997). The neural code between neocortical pyramidal neurons depends on neurotransmitter release probability. *Proc. Natl. Acad. Sci. USA 94*, 719–723.

van Vreeswijk, C., & Sompolinsky, H. (1996). Chaos in neuronal networks with balanced excitatory and inhibitory activity. *Science 274*, 1724–1726.

Vogels, T. P., & Abbott, L. F. (2005). Signal propagation and logic gating in networks of integrate-and-fire neurons. *J. Neurosci. 25*(46), 10786–10795.

Vogelstein, R., Mallik, U., Vogelstein, J., & Cauwenberghs, G. (2007). Dynamically reconfigurable silicon array of spiking neurons with conductance-based synapses. *IEEE Trans. Neural Netw. 18*(1), 253–265.

Wilson, R. I., & Laurent, G. (2005). Role of GABAergic inhibition in shaping odor-evoked spatiotemporal patterns in the drosophila antennal lobe. *J. Neurosci. 25*(40), 9069–9079.

Yazdanbakhsh, A., Babadi, B., Rouhani, S., Arabzadeh, E., & Abbassian, A. (2002). New attractor states for synchronous activity in synfire chains with excitatory and inhibitory coupling. *Biol. Cybern. 86*, 367–378.